

SQL (Basic)

Vanessa Casillas

Graduate Fellow

Before we start

- Download
 - DataGrip by JetBrains environment
 - <https://www.jetbrains.com/datagrip/>
 - Download and install on local machine (available for Windows and Mac)
- Survey
- <https://github.com/CMC-QCL/SQL>

Agenda

SQL Overview

Relational Databases

Databases

Basic SQL Statements

SQL Database Query Difference

DataGrip

Basics SQL Commands Hands-on

Operation Order

Resources

Contact Info

If you want
to...

See data

Relationship between data

Find any cell

Add data to cells

To add order to information

SQL Overview

Structured Query
Language

Crud: create, read,
update, delete

- Create databases and tables
- Look at specific data
- Make changes to data and remove data
- And more (level 2)

Relational Databases

A collection of tables

- Set of columns in each table and rows with data
- Each row is called a **record**
- Data between tables can be related between each other

Collection of tables is
called **schema**

Relational DBMS

Courses_Students

ID	ClassID	Semester
71225	1005	Fall21
86634	1006	Spr22
32238	1009	Spr22

School_Courses

ClassID	Title	ClassNum
1005	Intro to Art History	500
1006	Intro to SQL	501
1009	Intro to Datebases	300

Students_Attendees

ID	Name	Grade	DOB
71225	Lili	Freshman	03/12/1995
32238	Brenda	Senior	05/28/1989
86634	James	Freshman	09/20/1998

The Relational Model

Primary Key

↓

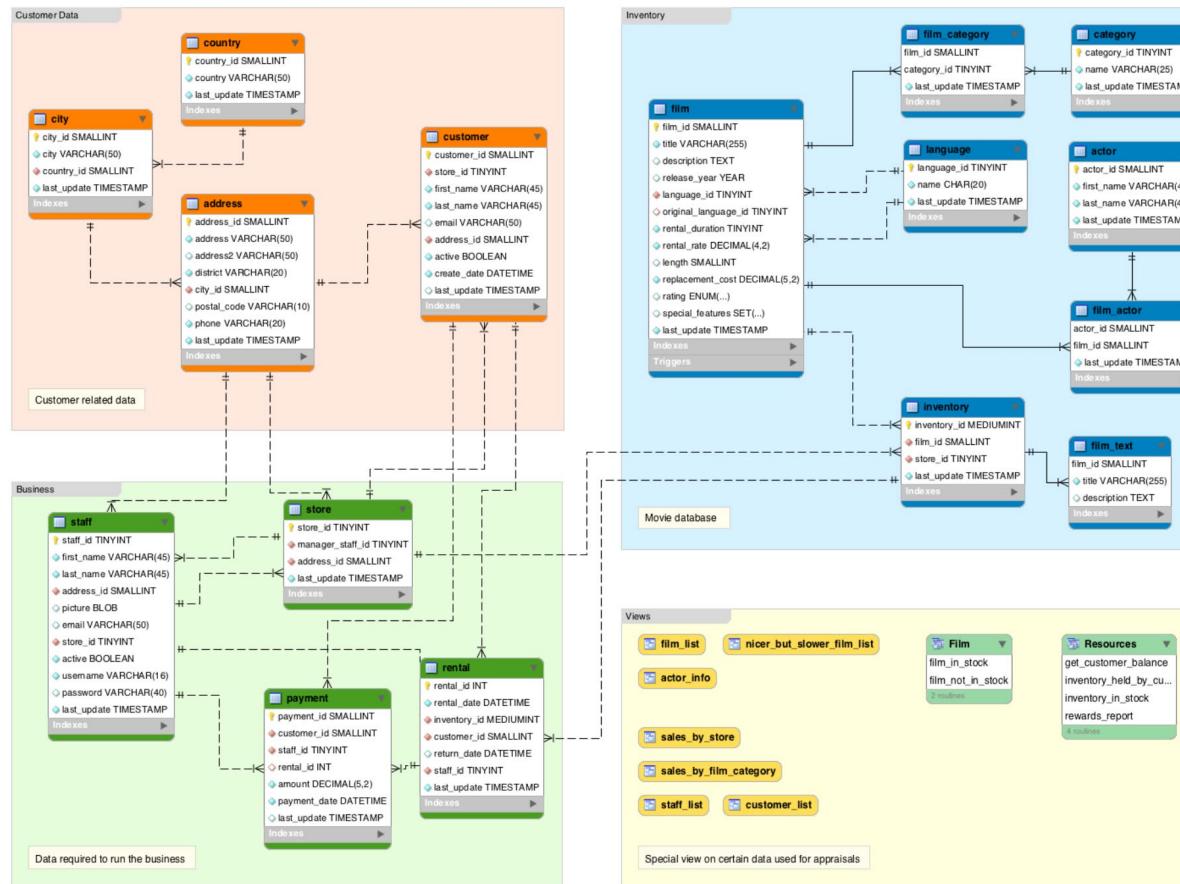
Row or Tuple →

CustomerID	CustomerName	Status
1	Chase	Active
2	Jackie	Active
3	Robert	Inactive

Rows are cardinality
Columns are degree
Table called a Relation

↑
Column or Attribute

Schema Example



<https://database.guide/what-is-a-database-schema/>

Vocabulary

- ▶ Data Definition Language (DDL):
 - ▶ CREATE, DROP, ALTER, TRUNCATE
- ▶ Data Manipulation Language (DML):
 - ▶ INSERT, UPDATE, DELETE
- ▶ Data Query Language (DQL):
 - ▶ SELECT, JOIN
- ▶ Data Control Language (DCL):
 - ▶ GRANT, REVOKE

Basic SQL Statements

The screenshot shows a DataGrip IDE interface. At the top, there's a code editor window titled "console_2" containing a SQL script. A red circle highlights the line "SELECT * FROM state_crime;". Below the code editor is a results table titled "Output" for the query "SELECT * FROM state_crime". The table has columns: State, Crime_Year, Population, Rates_Property_Theft, and Rates_Violent. The data shows crime statistics for various US states in 2019.

State	Crime_Year	Population	Rates_Property_Theft	Rates_Violent
Alabama	2019	4903185	531.9	
Alaska	2019	731545	487.1	
Arizona	2019	7278717	394.3	
Arkansas	2019	3017804	599.6	
California	2019	3951223	386.1	
Colorado	2019	5758736	348.4	
Connecticut	2019	3565287	180.7	
Delaware	2019	973764	304.8	



VIEW: virtual table

The screenshot shows a DataGrip IDE interface. At the top, there's a code editor window titled "state_computer_data.insertsql" containing a complex SQL query. A red arrow points from the line "SELECT * FROM state_computer_data;" in the code editor down to the "Operations Tree" in the results panel below. The results panel shows the execution plan for the query, which includes a Hash Join (inner hash) operation. The operations tree details the execution steps and their costs.

Operation	Params	Rows	Total Cost	Startup Cost	Raw Desc
Select		796	801.2	801.2	(state_computer_data-<hash>(state_computer_data))
Filter		796	0.04	0.04	Full Scan (Table scan: state_crime)
Hash Unique (Hash)		51	5.35	5.35	Full Scan (Table: state_computer_data)

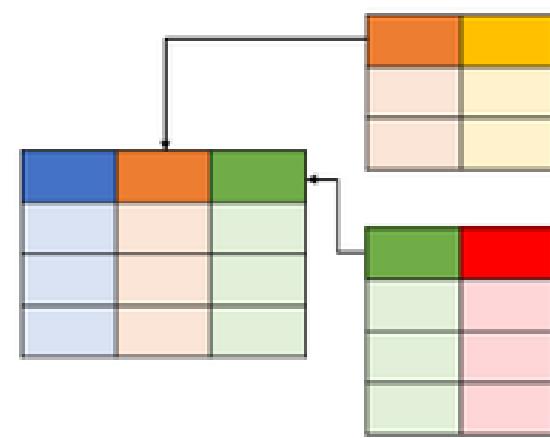


INDEX: users cannot see the index; it is used to retrieve data from the database more quickly

Relational Database management system

ACID compliant:
(Atomicity, Consistency, Isolation, and Durability)

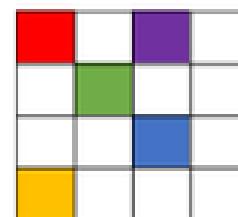
SQL DATABASES



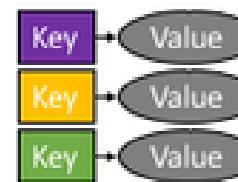
Relational

Vertically Scalable
Fixed or predefined Schema
Used for complex queries

NoSQL DATABASES



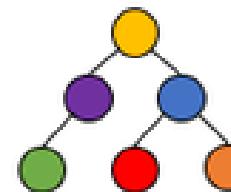
Column



Key-Value



Graph



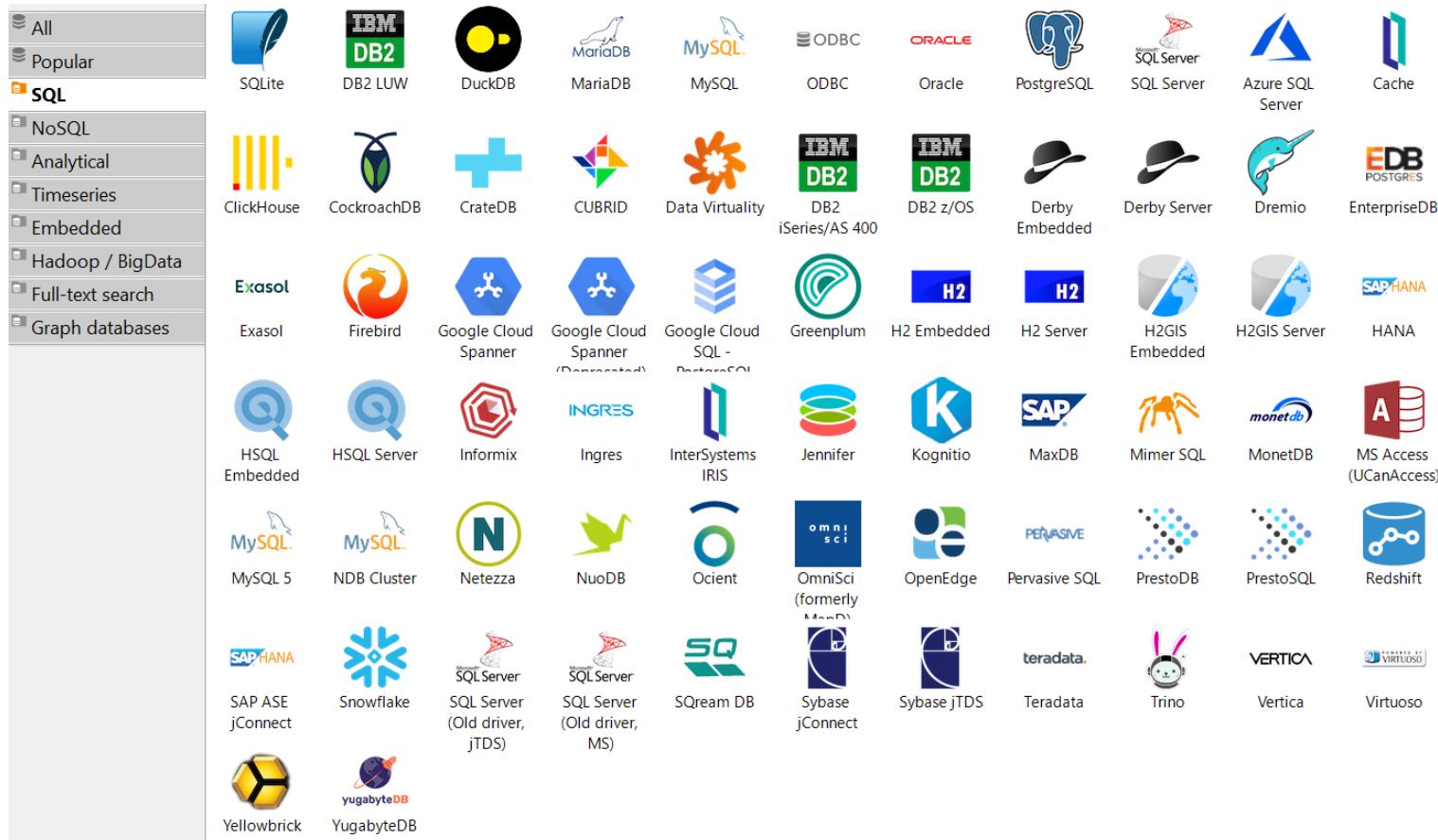
Document

Dynamic Scalable
Dynamic Schema
Not good for complex queries

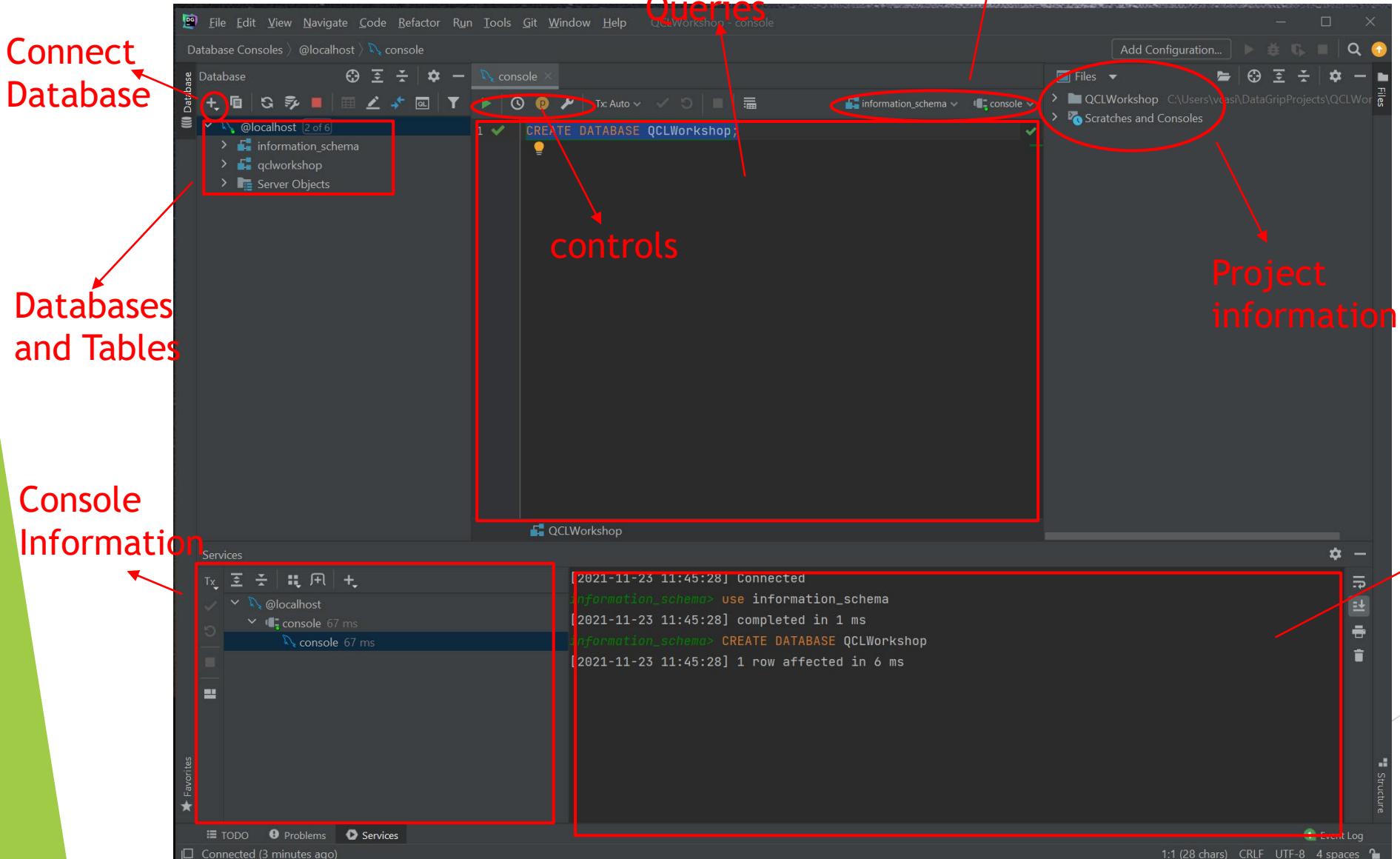
Distributed Database Management system

CAP Theorem:
Consistency, Availability, Partitioning

Relational SQL Databases



Interface



Today's Goals



Whole - Big Picture

- ▶ Be able to take any data set and import into your database
- ▶ Be able to sort and look through your data set
- ▶ Be able to create your own data set and input data
- ▶ Be able to join data sets together



Parts - Small angles

- ▶ Use basic SQL commands
- ▶ Understand the process of writing a query
- ▶ Understand how data can be joined
- ▶ Understand the behind the scenes of how the query is operating

Today's Data



- ▶ Modified datasets for workshop (4 files total)
- ▶ **State Crime CSV File**
 - ▶ `state_crime.csv`
 - ▶ information on the crime rates and totals for states across the United States for a wide range of years
 - ▶ reports go from 1960 to 2019 (only used 2010, 2014 and 2019)
 - ▶ https://corgis-edu.github.io/corgis/csv/state_crime/
- ▶ **State Demographics CSV and SQL Files**
 - ▶ `state_computer_data.sql`, `state_workforce.csv`, `state_people.sql`
 - ▶ summarized information obtained about states in the United States from 2015 through 2019 through the United States Census Bureau
 - ▶ just the summarized data as of 2019
 - ▶ https://corgis-edu.github.io/corgis/csv/state_demographics/

<https://corgis-edu.github.io/corgis/>

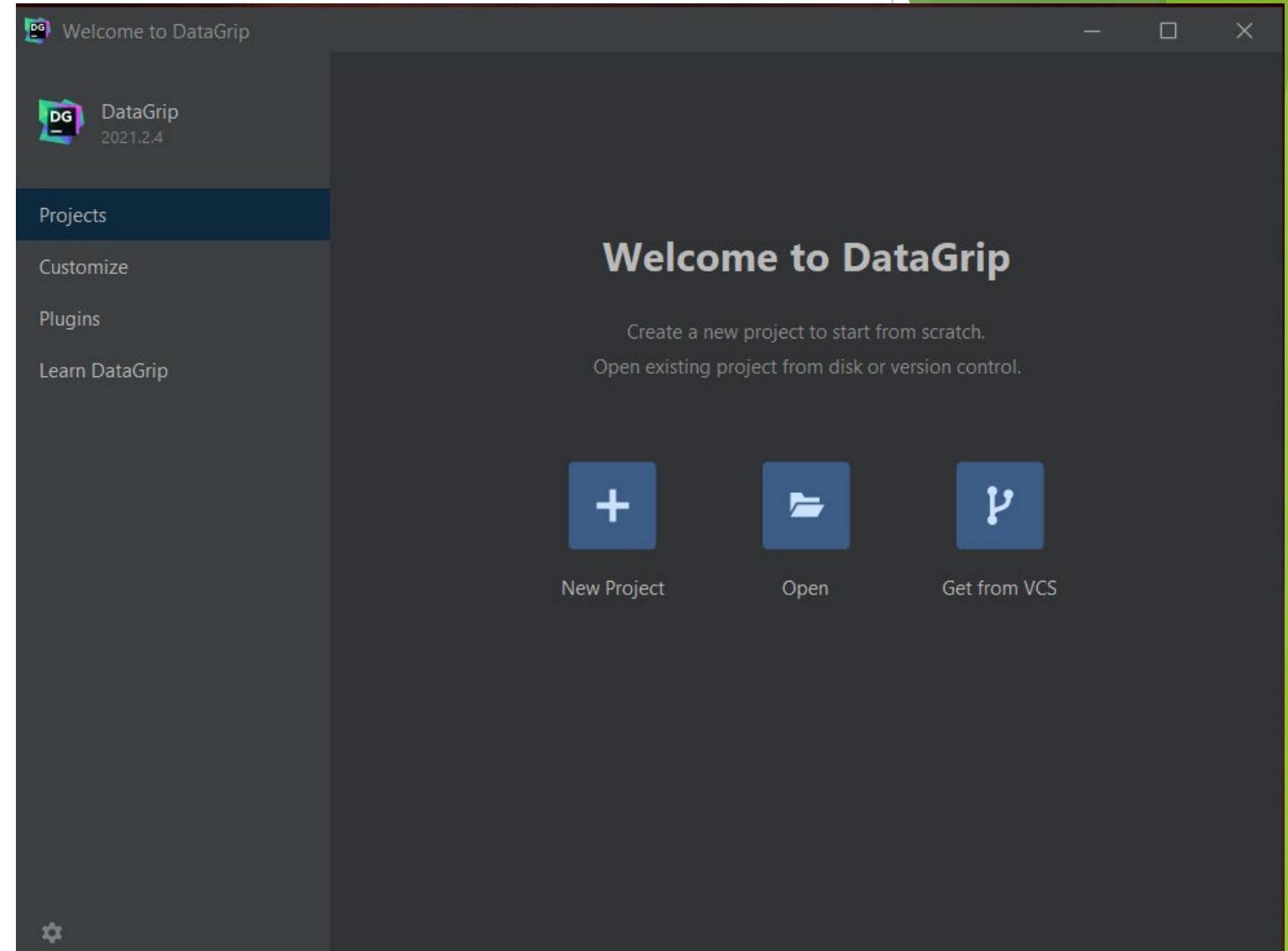
Basic SQL Commands

Hands-on

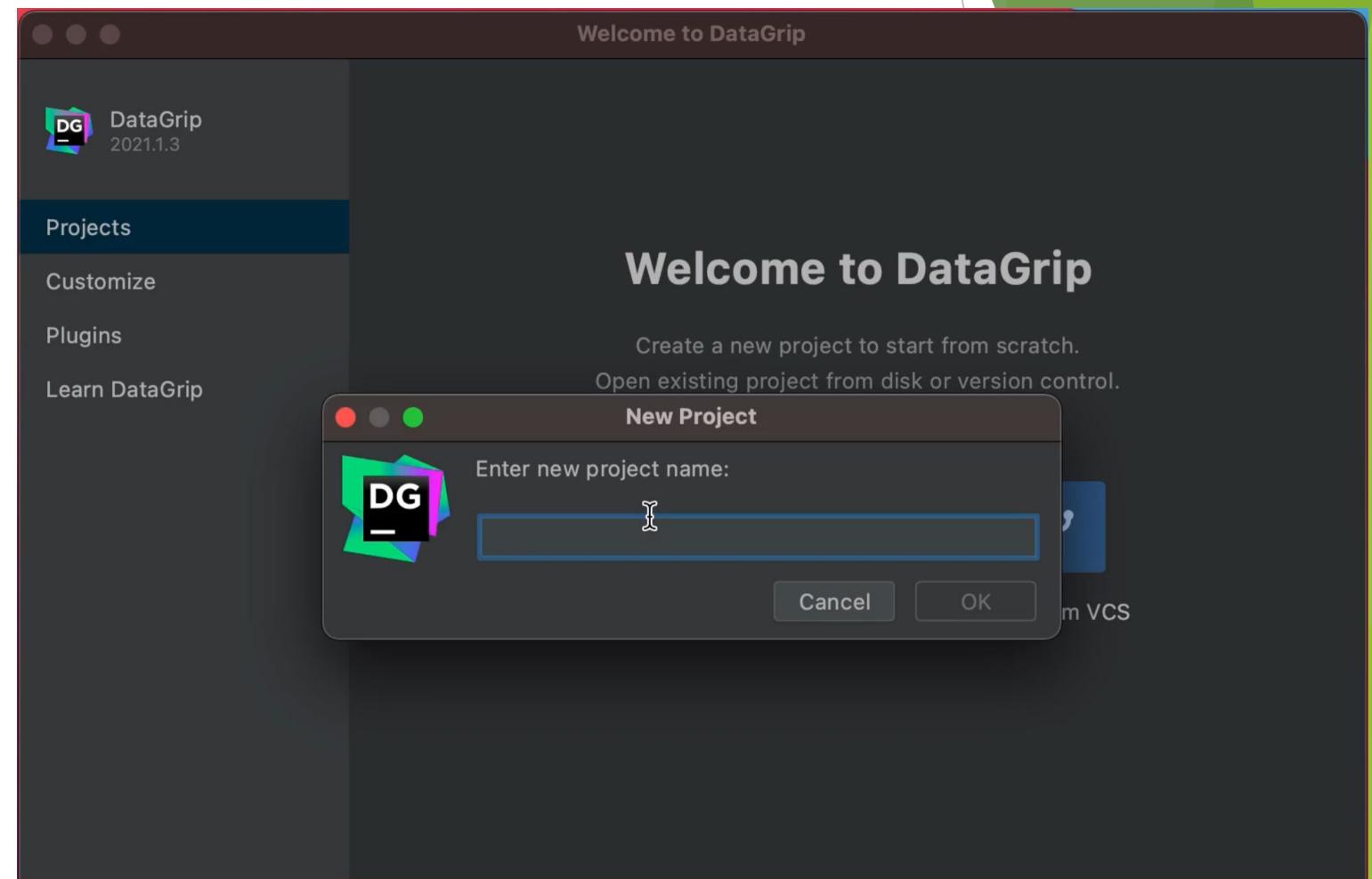
Hands-on Agenda

- ▶ Connect to Sever MySQL
- ▶ Start a Database called “QCLWorkshop”
- ▶ Import a data table called **state_crime**
- ▶ Query using the **SELECT** statement with *
- ▶ Filter a query using the **WHERE** clause and **ORDER BY**
- ▶ Create data table called **state_computer_data**
- ▶ Insert **state_computer_data** from **sql file**
- ▶ Relate the two tables using the **JOIN** connector make **Alias**
- ▶ Use **Group by** to look sort the data set
- ▶ Throughout the workshop: Activities involving files named **state_workforce** and **state_people**

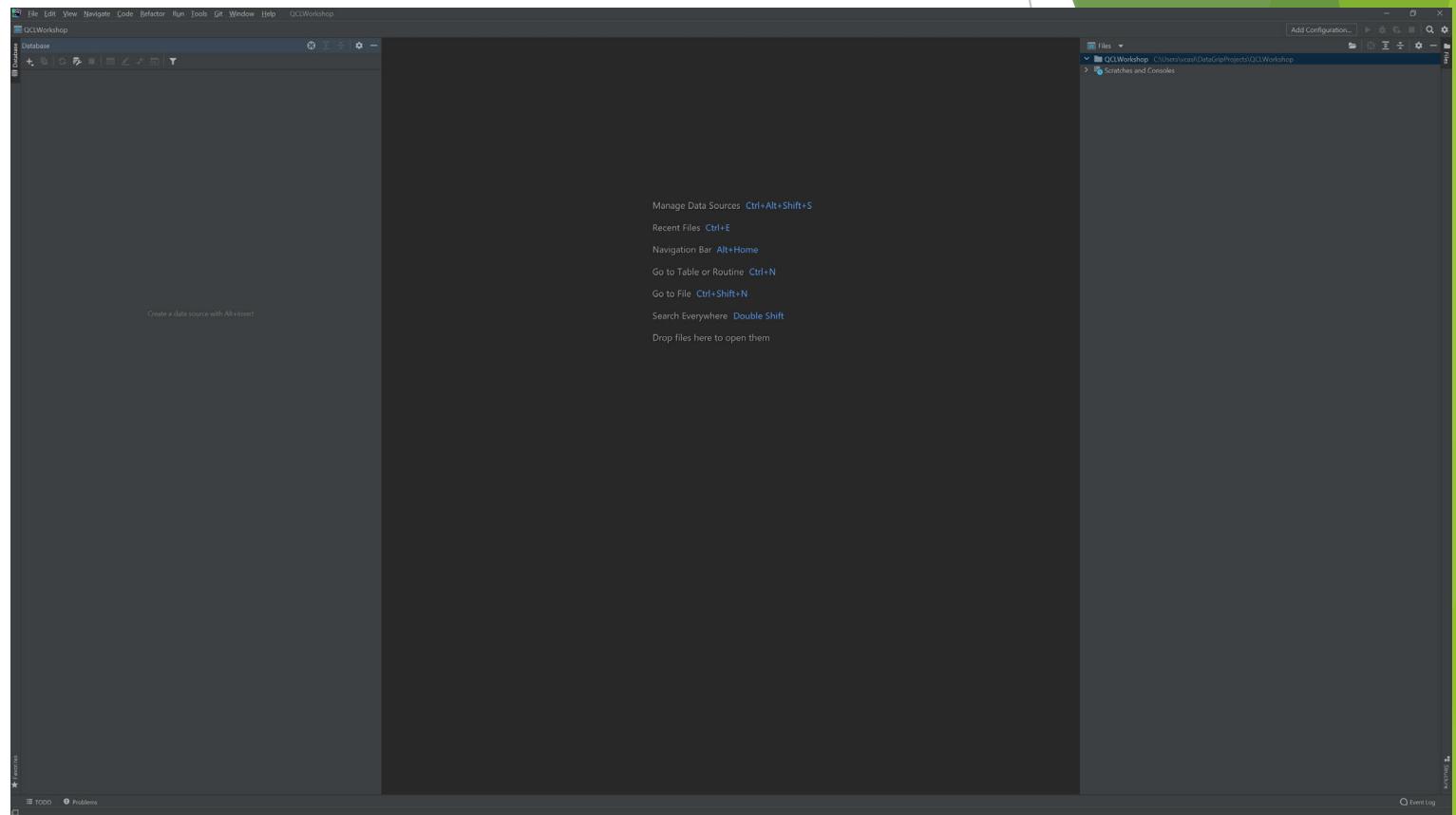
Click New Project



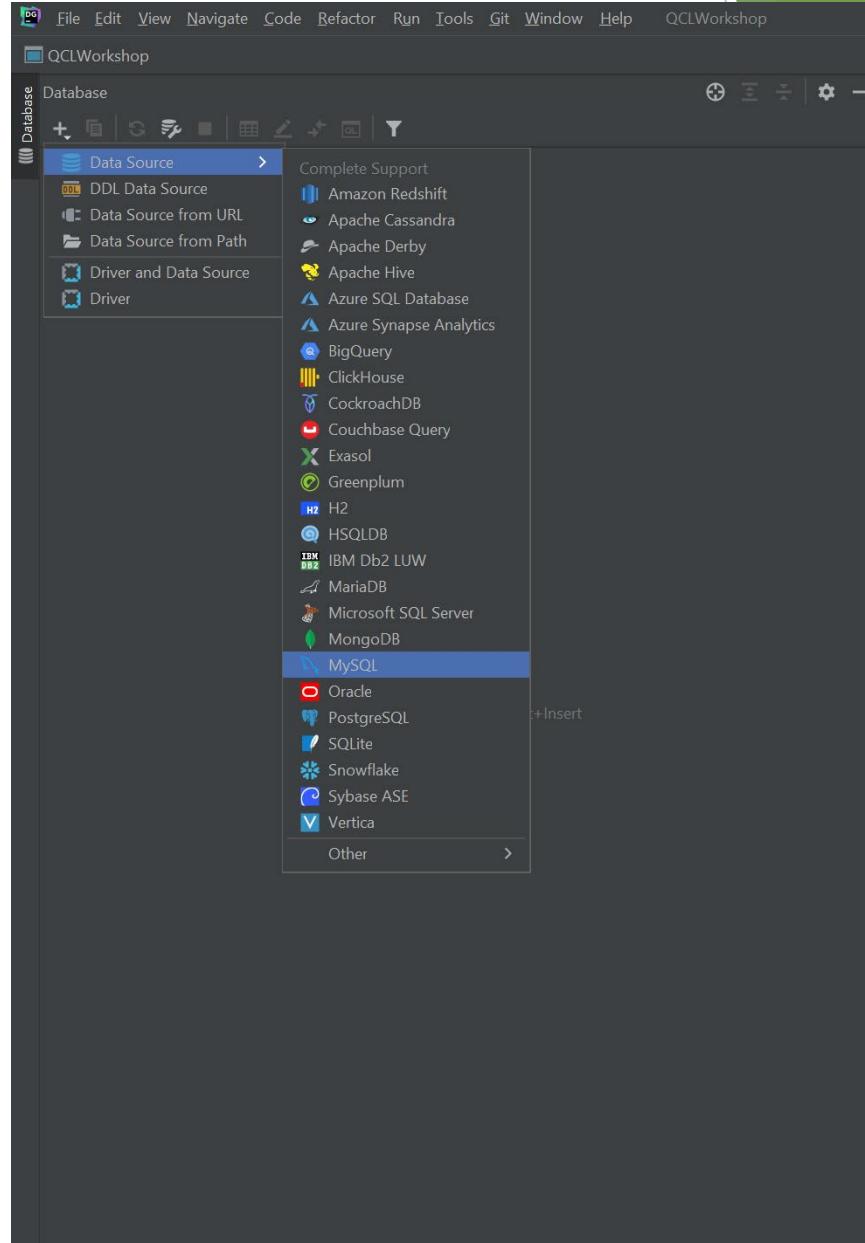
Make new
project called
“QCLWorkshop”



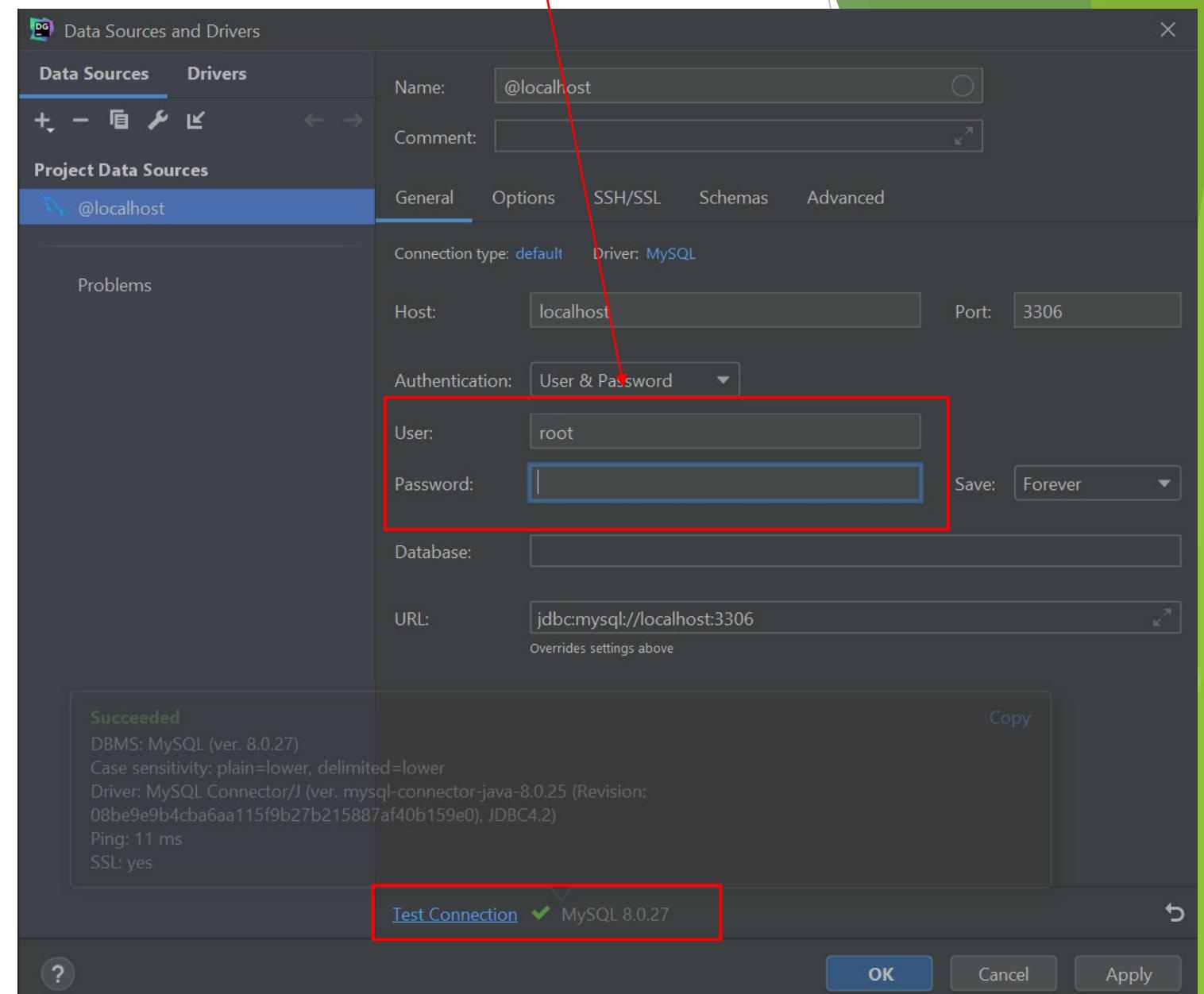
What is look like when you have it setup



Click on the
Plus Sign
Click on Data
Source
Click on MySQL

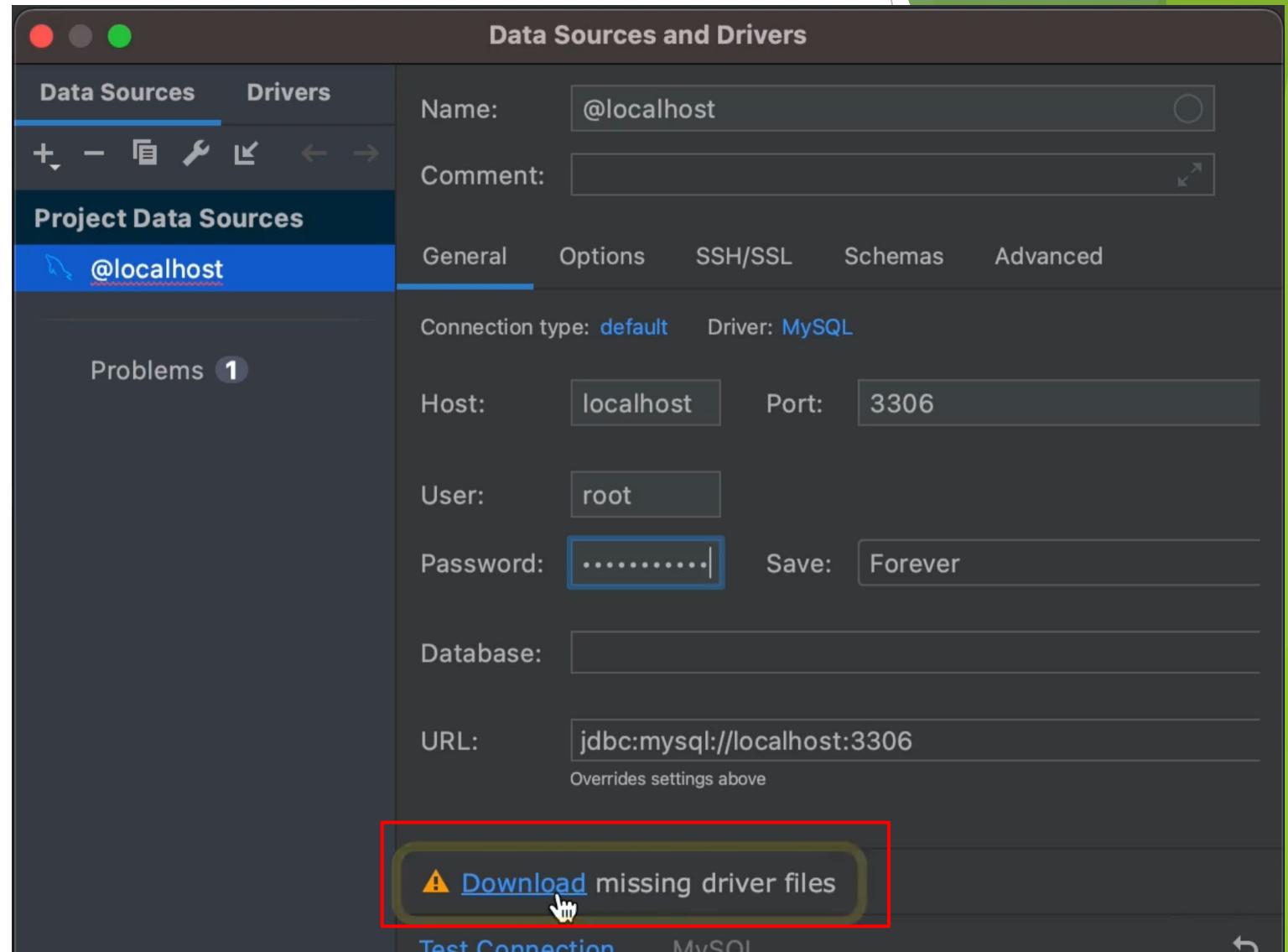


Password that you made or “1234pass”

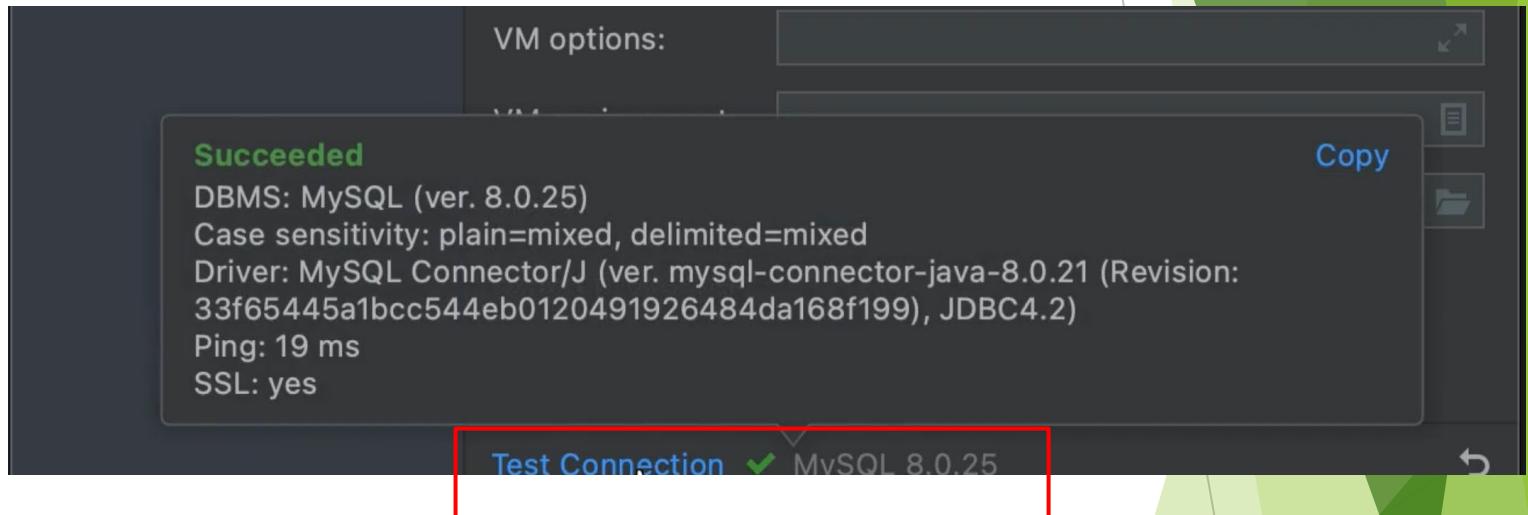


Fill in User: root
Fill in Password:
1234pass
Test Connection

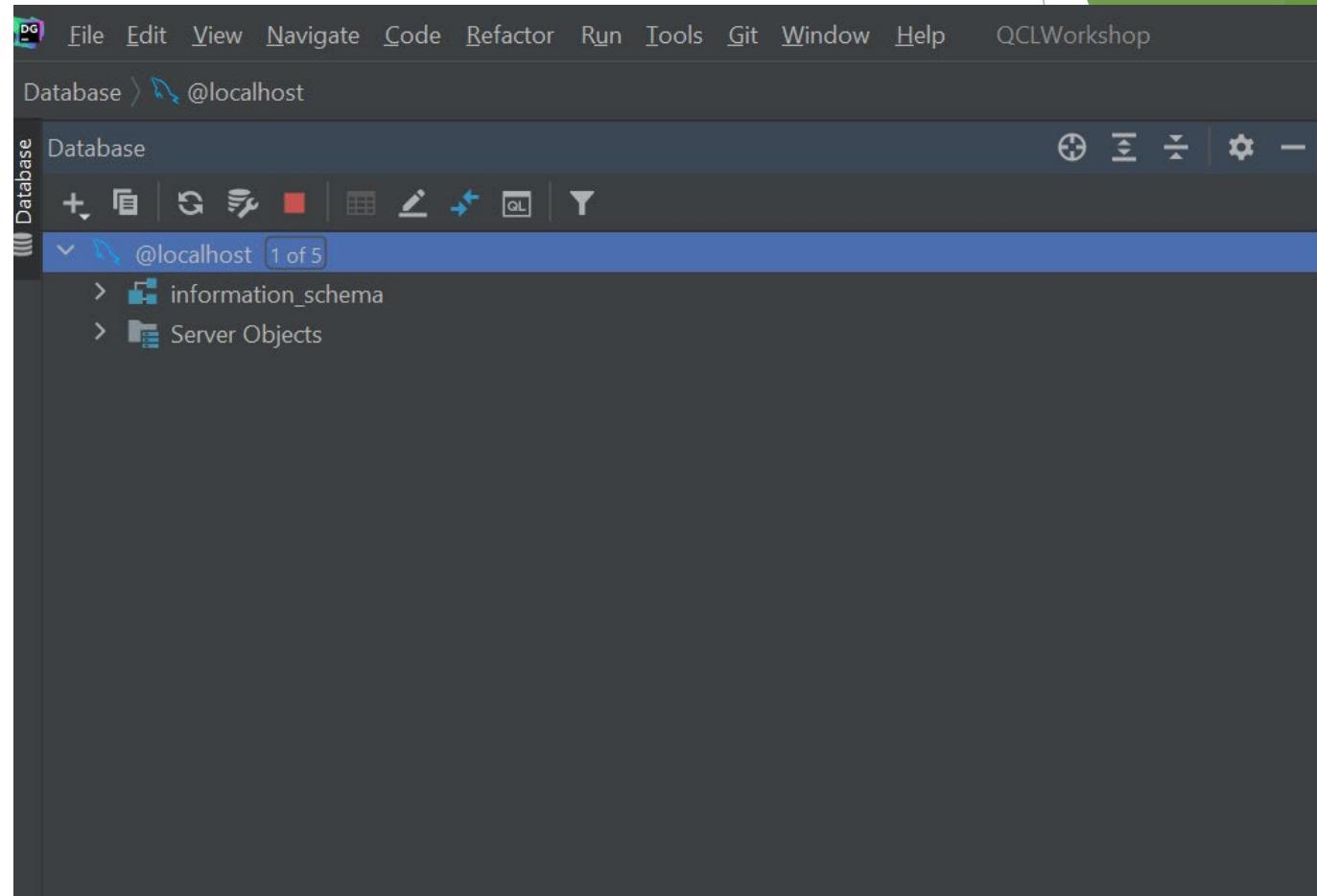
Download missing driver files



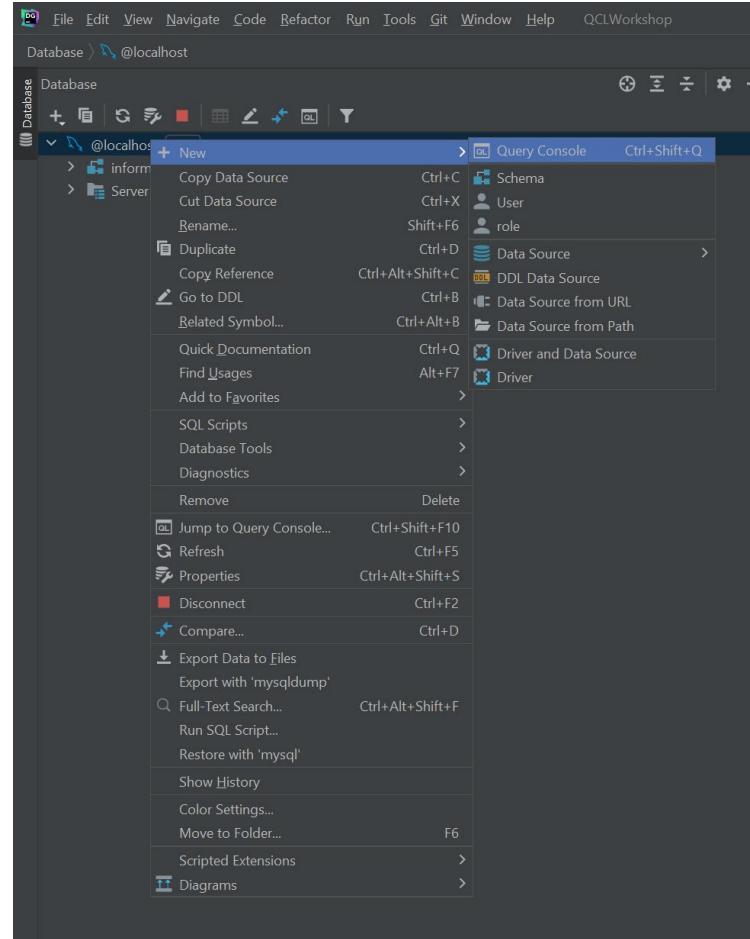
Test Connection

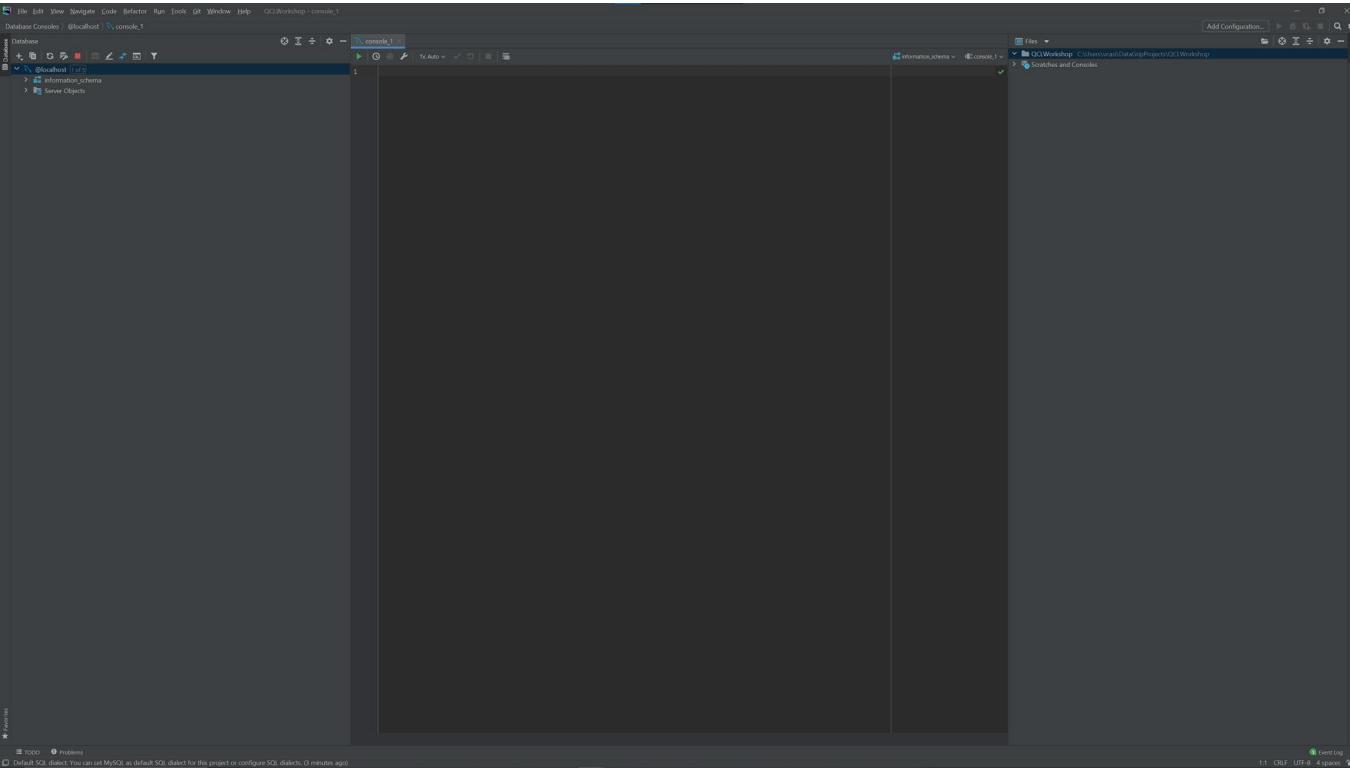


See the
Server on
the left-
hand side



Right click,
click new,
click Query
Console





You are ready for the workshop

The screenshot shows the DataGrip IDE interface. The main window displays a SQL console titled "console_1" connected to the database "localhost". The code editor contains the following SQL script:

```
/* [QCL WORKSHOP] SQL (Basic)
 * Hands-on
 * by Vanessa Casillas (Graduate Fellow)
 */
/* WARNING: MAKE SURE TO CHECK NAMES OF TABLES,
 * THEY CAN BE A DIFFERENT NAME ON YOUR COMPUTER! */
```

The status bar at the bottom shows a warning message: "[42000][1049] Unknown database 'qclworkshop'. (a minute ago)". The right side of the interface shows a file browser with the path "QCLWorkshop C:\Users\vcasi\DataGripProjects\QCLWorkshop\Scratches and Consoles".

Note taking in SQL

Notetaking

```
1  /* [QCL WORKSHOP] SQL (Basic)           ✓ 1 ^ v
2   * Hands-on
3   * by Vanessa Casillas (Graduate Fellow)
4   */
5
6  /* WARNING: MAKE SURE TO CHECK NAMES OF TABLES,
7   * THEY CAN BE A DIFFERENT NAME ON YOUR COMPUTER! */
```

The screenshot shows the DataGrip IDE interface with the following details:

- Top Bar:** File, Edit, View, Navigate, Code, Refactor, Run, Tools, Git, Window, Help, QCLWorkshop - console_2
- Database Tree:** Database, @localhost (selected), information_schema, qcworkshop, Server Objects.
- Console:** Database Consoles > @localhost > console_2. The code editor contains the following SQL script:

```
1 /* [QCL WORKSHOP] SQL (Basic)
2 * Hands-on
3 * by Vanessa Casillas (Graduate Fellow)
4 */
5
6 /* WARNING: MAKE SURE TO CHECK NAMES OF TABLES,
7 * THEY CAN BE A DIFFERENT NAME ON YOUR COMPUTER! */
8
9
10 /*Create a database*/
11 CREATE DATABASE QCLWorkshop;
```
- Services Panel:** Shows a connection to @localhost, console_2, completed in 44 ms. The log output is:

```
[2021-11-23 16:28:11] Connected
information_schema> use information_schema
[2021-11-23 16:28:11] completed in 1 ms
information_schema> CREATE DATABASE QCLWorkshop
[2021-11-23 16:28:11] 1 row affected in 5 ms
```
- Bottom Status Bar:** TODO, Problems, Services, Connected (a minute ago), Event Log, 11:1 (28 chars), CRLF, UTF-8, 4 spaces.

Creating a Database

Calling the Database

The screenshot shows a DataGrip IDE interface with a central code editor window titled "console_1". The code editor displays the following SQL script:

```
/* [QCL WORKSHOP] SQL (Basic)
 * Hands-on
 * by Vanessa Casillas (Graduate Fellow)
 */
/* WARNING: MAKE SURE TO CHECK NAMES OF TABLES,
 * THEY CAN BE A DIFFERENT NAME ON YOUR COMPUTER! */

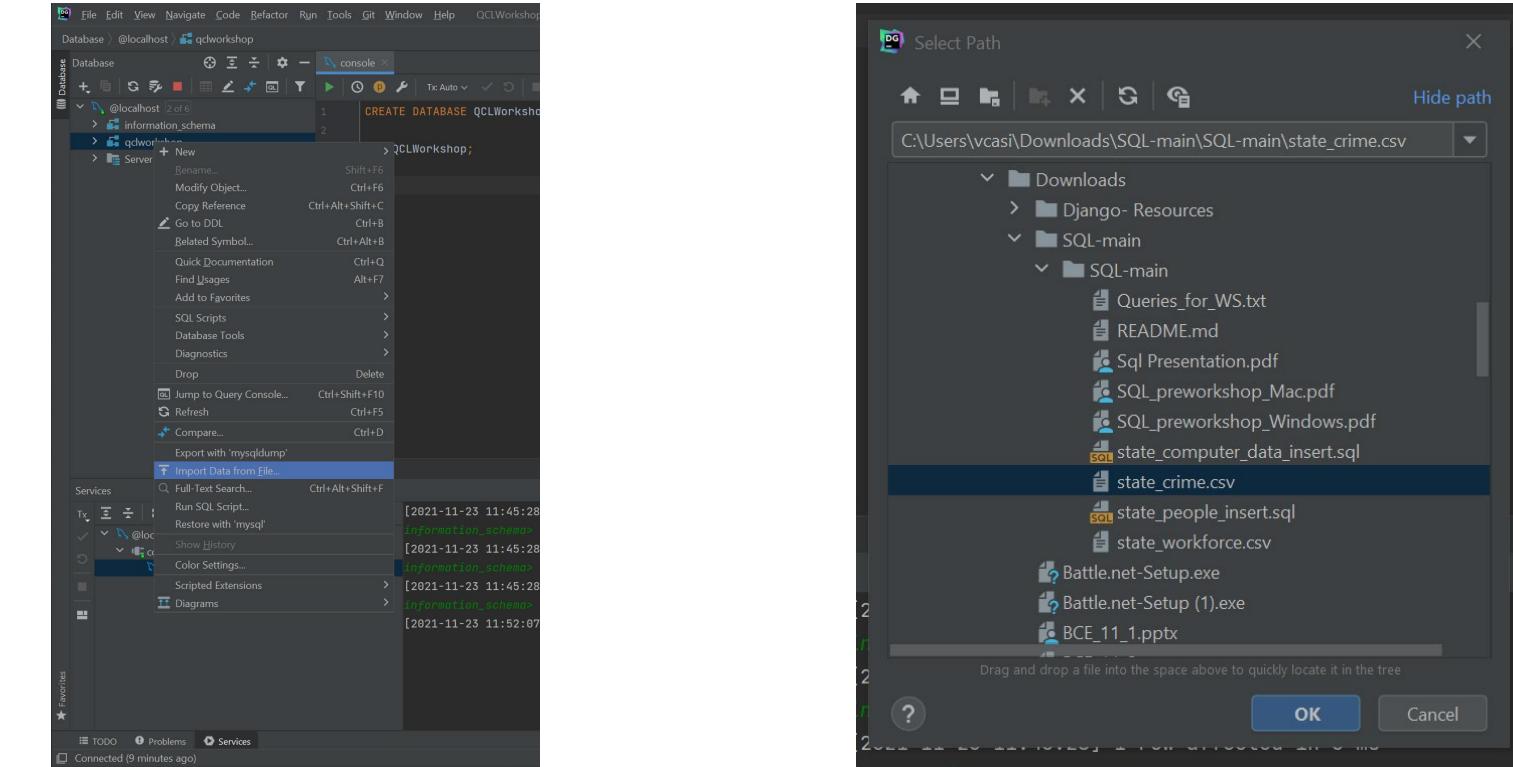
/*Create a database*/
CREATE DATABASE QCLWorkshop;

/*Calling the database*/
USE QCLWorkshop;
```

The "Services" panel at the bottom shows a transaction named "console_1" with a duration of 7 ms. The transaction log shows the following activity:

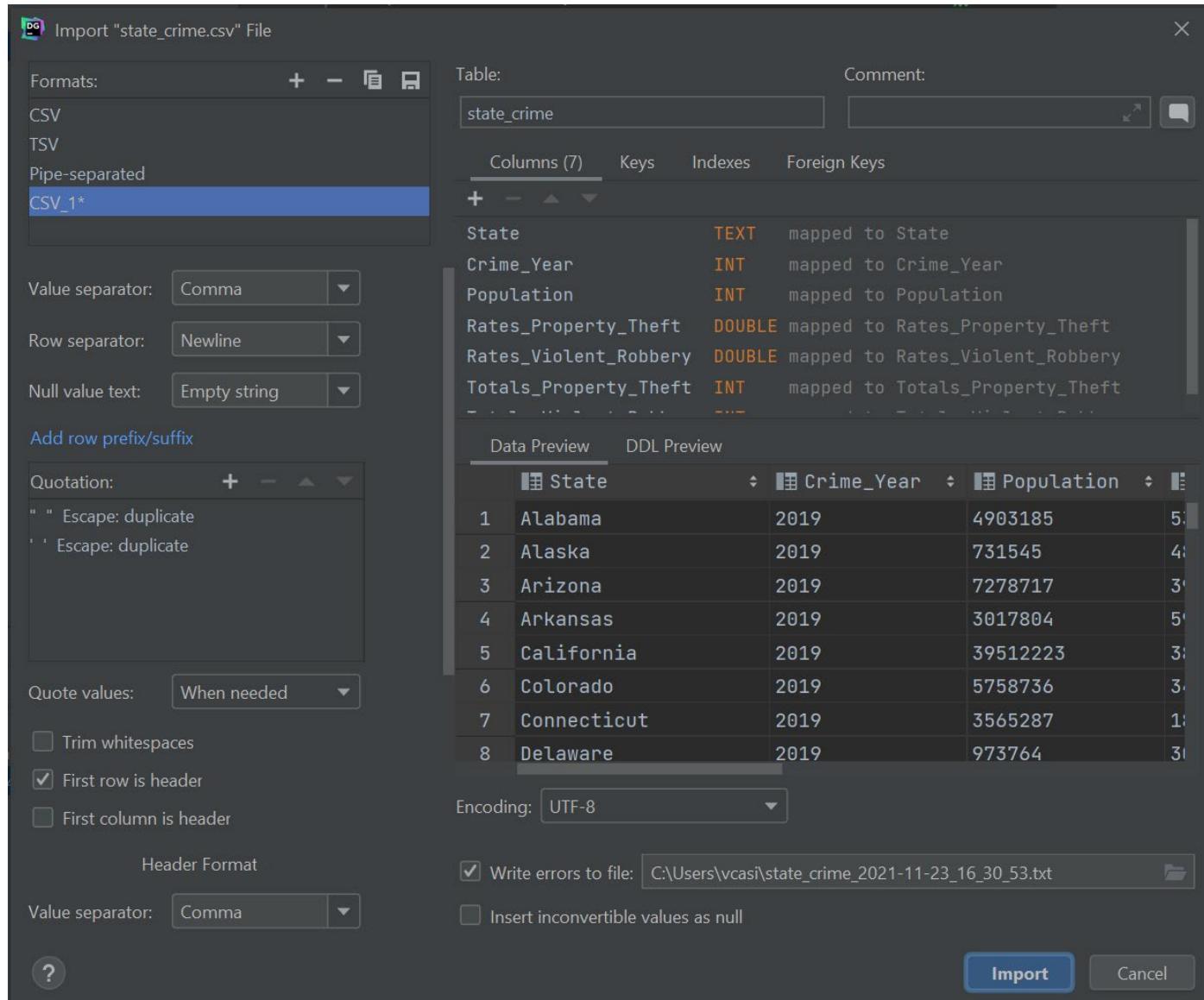
- [2021-11-23 12:10:03] Connected
- information_schema> use information_schema
- [2021-11-23 12:10:03] completed in 1 ms
- information_schema> CREATE DATABASE QCLWorkshop
- [2021-11-23 12:10:03] 1 row affected in 3 ms
- information_schema> USE QCLWorkshop
- [2021-11-23 12:10:51] completed in 1 ms

The status bar at the bottom indicates "Connected (2 minutes ago)" and "Event Log".



Import CSV

Vanessa Casillas (Graduate Fellow at QCL)



Import CSV (cont.)

Data types

- ▶ Text vs. Varchar
- ▶ Int vs. Integer
- ▶ Double vs. Real vs. Float

The screenshot shows the DataGrip IDE interface. The top navigation bar includes File, Edit, View, Navigate, Code, Refactor, Run, Tools, Git, Window, Help, and QCLWorkshop - console_2. The left sidebar displays the Database tree, showing @localhost (2 of 6) with information_schema and qclworkshop databases, and qclworkshop containing tables (1) with state_crime selected. Below it, the Services panel shows a transaction (Tx) for @localhost with a connection to default (75 ms) and another to console_2 (14 ms). The main central area is a code editor for 'console_2' with the following SQL script:

```
/* [QCL WORKSHOP] SQL (Basic)
 * Hands-on
 * by Vanessa Casillas (Graduate Fellow)
 */
/* WARNING: MAKE SURE TO CHECK NAMES OF TABLES,
 * THEY CAN BE A DIFFERENT NAME ON YOUR COMPUTER! */

/*Create a database*/
CREATE DATABASE QCLWorkshop;

/*Calling the database*/
USE QCLWorkshop;
```

The bottom right corner of the code editor shows a message: '@localhost state_crime.csv imported to state_crime: 156 rows (17 ms)'. The bottom status bar indicates the connection is synchronized at 267 ms.

Successful Import of CSV

Writing Query

Select - Returns the data that was requested

From - choose a table to draw information from

Join - matches records from different tables

Where - filters data bases on request

Group By - aggregates the data

Having - filers aggregated data

Order by - sorts the data

Limit - limit the number of rows returned

Select with *

```
/* Select Statement with wildcard */  
SELECT * FROM state_crime;
```

Detailed view

```
SELECT column1, column2, ...  
FROM table_name;
```

`*`: wildcard (select all)

Table name

`SELECT * FROM state_crime;` → syntax

What do you want to see?

what table?

```
graph TD; A["* : wildcard (select all)"] -- green --> B["Table name"]; C["SELECT * FROM state_crime;"] -- purple --> D["Table name"]; E["what table?"] -- red --> F["What do you want to see?"]
```

File Edit View Navigate Code Refactor Run Tools Git Window Help QCLWorkshop - console_2

Database Consoles > @localhost > console_2

Database

console_2

1 /* [QCL WORKSHOP] SQL (Basic)

2 * Hands-on

3 * by Vanessa Casillas (Graduate Fellow)

4 */

5

6 /* WARNING: MAKE SURE TO CHECK NAMES OF TABLES,

7 * THEY CAN BE A DIFFERENT NAME ON YOUR COMPUTER! */

8

9

10 /*Create a database*/

11 CREATE DATABASE QCLWorkshop;

12

13 /*Calling the database*/

14 USE QCLWorkshop;

15

16 /*Select Statement with wildcard */

17 SELECT * FROM state_crime;

Services

Tx

Output

qdworkshop.state_crime

1 State Crime_Year Population Rates_Property_Theft Rates_Violent

2 Alabama 2019 4903185 531.9

3 Alaska 2019 731545 487.1

4 Arizona 2019 7278717 394.3

5 Arkansas 2019 3017804 599.6

6 California 2019 39512223 386.1

7 Colorado 2019 5758736 348.4

8 Connecticut 2019 3565287 180.7

9 Delaware 2019 973764 304.8

Event Log

@localhost: qdworkshop synchronized (267 ms) (a minute ago)

17:1 (26 chars) CRLF UTF-8 4 spaces

Select with columns

```
/* Select Statement */
SELECT Population,
       Totals_Property_Theft,
       Totals_Violent_Robbery
  FROM state_crime;
```

Detailed view

```
SELECT column1, column2, ...  
FROM table_name;
```

What do you want to see?

Column names

```
SELECT Population, Totals_Property_Theft, Totals_Violent_Robbery  
FROM state_crime; —→ syntax
```

what table?

Table name

The screenshot shows the DataGrip IDE interface. The top navigation bar includes File, Edit, View, Navigate, Code, Refactor, Run, Tools, Git, Window, Help, and QCLWorkshop - console_2. The left sidebar displays the Database tree, showing @localhost (2 of 6) with information_schema, qcworkshop, and tables. Under qcworkshop, there is a tables folder containing state_crime, which has columns State, Crime_Year, Population, Rates_Property_Theft, Rates_Violent_Robbery, Totals_Property_Theft, and Totals_Violent_Robbery. The main editor window contains the following SQL code:

```
/* WARNING: MAKE SURE TO CHECK NAMES OF TABLES,  
 * THEY CAN BE A DIFFERENT NAME ON YOUR COMPUTER! */  
  
/*Create a database*/  
CREATE DATABASE QCLWorkshop;  
  
/*Calling the database*/  
USE QCLWorkshop;  
  
/* Select Statement with wildcard */  
SELECT * FROM state_crime;  
  
/* Select Statement */  
SELECT Population,  
    Totals_Property_Theft,  
    Totals_Violent_Robbery  
FROM state_crime;
```

The Services panel at the bottom shows a transaction list with @localhost/default (75 ms), @localhost/console_2 (51 ms), and the current selected transaction @localhost/console_2 (51 ms). The Output tab displays the results of the SELECT query:

	Population	Totals_Property_Theft	Totals_Violent_Robbery
1	4903185	26079	3941
2	731545	3563	826
3	7278717	28699	6410
4	3017804	18095	1557
5	39512223	152555	52301
6	5758736	20064	3663
7	3565287	6441	1929
8	973764	2968	790

The status bar at the bottom indicates 156 rows retrieved starting from 1 in 45 ms (execution: 2 ms, fetching: 43 ms).



Activity #1 - Select

- ▶ Import the file named state_workforce
 1. How many rows and attributes does this table have?

Where Clause

```
/* Where Clause */
SELECT Population,
       Totals_Property_Theft,
       Totals_Violent_Robbery
  FROM state_crime
 WHERE Totals_Violent_Robbery >=3000;
```

Detailed view

```
SELECT column1, column2, ...
FROM table_name
WHERE condition;
```

What do you want to see?

Column names

what table?

```
SELECT Population, Totals_Property_Theft, Totals_Violent_Robbery
FROM state_crime
WHERE Totals_Violent_Robbery >=3000;
```

Clause: to filter

Condition

The screenshot shows the DataGrip IDE interface with the following components:

- Top Bar:** File, Edit, View, Navigate, Code, Refactor, Run, Tools, Git, Window, Help, QCLWorkshop - console_2
- Database Explorer:** Shows the database structure at `@localhost [2 of 6]`. It includes the `information_schema`, `qdworkshop` schema (with `tables 1` containing `state_crime`), and `Server Objects`.
- Code Editor:** A tab titled `console_2` contains the following SQL code:

```
/*Calling the database*/
USE QCLWorkshop;

/* Select Statement with wildcard */
SELECT * FROM state_crime;

/* Select Statement */
SELECT Population,
       Totals_Property_Theft,
       Totals_Violent_Robbery
FROM state_crime;

/* Where Clause */
SELECT Population,
       Totals_Property_Theft,
       Totals_Violent_Robbery
FROM state_crime
WHERE Totals_Violent_Robbery >=3000;
```
- Services:** Shows the transaction status for `@localhost` with `default` and `console_2` sessions.
- Output Tab:** Displays the results of the last query in a table format:

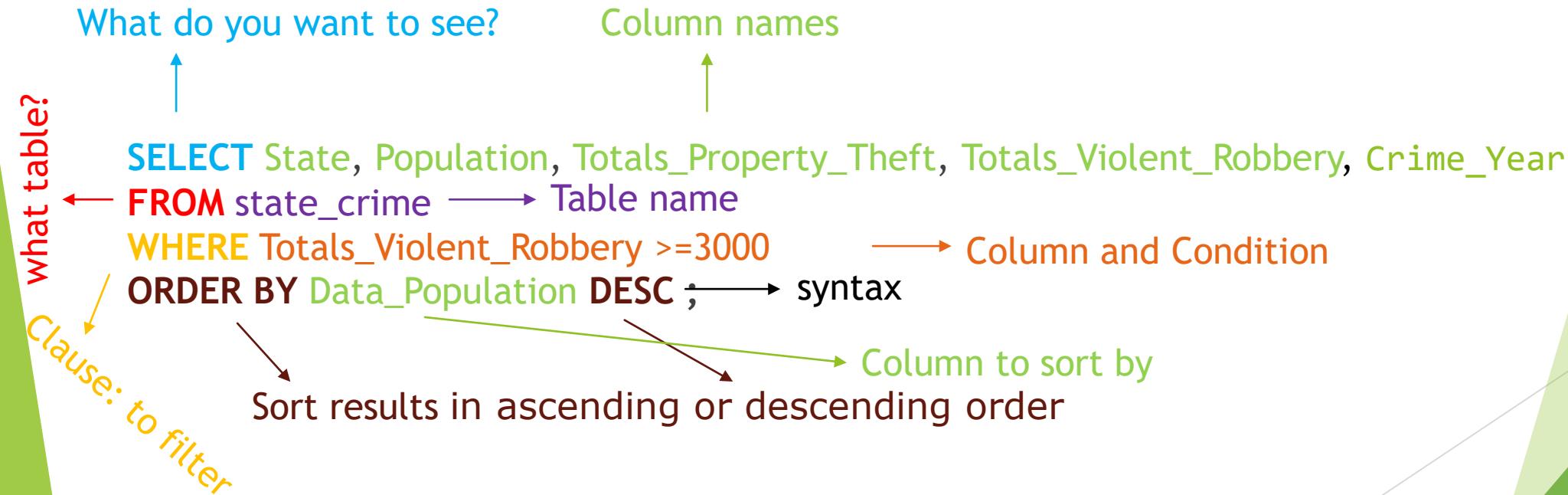
	Population	Totals_Property_Theft	Totals_Violent_Robbery
1	4903185	26079	3941
2	7278717	28699	6410
3	39512223	152555	52301
4	5758736	20064	3663
5	21477737	63396	16217
6	10617423	39506	7961
7	12671821	34433	12464
8	6732219	21795	5331
- Bottom Status Bar:** Shows "65 rows retrieved starting from 1 in 48 ms (execution: 2 ms, fetching: 46 ms)" and "26:1 (118 chars, 4 line breaks) CRLF UTF-8 4 spaces".

Order by

```
/* Order By */
SELECT State,
Population,
Totals_Property_Theft,
Totals_Violent_Robbery,
Crime_Year
FROM state_crime
WHERE Totals_Violent_Robbery >=3000
ORDER BY Population DESC;
```

Detailed view

```
SELECT column1, column2, ...
FROM table_name
WHERE condition;
ORDER BY column1, column2, ... ASC|DESC;
```



The screenshot shows the DataGrip IDE interface with the following components:

- Database Browser:** On the left, it displays the database structure. Under the `@localhost` connection, there are two databases: `information_schema` and `qclworkshop`. The `qclworkshop` database contains two tables: `state_computer_data` and `state_crime`. The `state_computer_data` table has four columns: `State` (varchar(20)), `Persons_per_Household` (float), `Households_with_computer` (float), and `Households_with_Internet` (float). The `state_crime` table has seven columns: `State` (text), `Crime_Year` (int), `Population` (int), `Rates_Property_Theft` (double), `Rates_Violent_Robbery` (double), `Totals_Property_Theft` (int), and `Totals_Violent_Robbery` (int).
- Query Editor:** The main area shows a SQL script named `state_computer_data_insert.sql` with the following content:

```
22     Totals_Violent_Robbery
23     FROM state_crime;
24
25     /* Where Clause */
26     SELECT Population,
27     Totals_Property_Theft,
28     Totals_Violent_Robbery
29     FROM state_crime
30     WHERE Totals_Violent_Robbery >=3000;
31
32     /* Order By */
33     SELECT State,
34     Population,
35     Totals_Property_Theft,
36     Totals_Violent_Robbery,
37     Crime_Year
38     FROM state_crime
39     WHERE Totals_Violent_Robbery >=3000
40     ORDER BY Population DESC;
```
- Services Panel:** At the bottom left, it shows the services running on the local host, with `console_2` currently selected.
- Output Viewer:** On the right, it displays the results of the query, which is a table with the following data:

	State	Population	Totals_Property_Theft	Totals_Violent_Robbery
1	United States	328239523	1117696	267988
2	United States	318857056	8277829	84041
3	United States	309330219	2168459	369089
4	California	39512223	152555	52301
5	California	38802500	947192	8398
6	California	37338198	228857	58116
7	Texas	28995881	113902	28988
8	Texas	26956958	813934	8236



Activity #2 - Where Clause and Order by

- ▶ Use where clause to find out how many people on average take longer than 20 mins to get to work?
- ▶ Use Order by to find out what state has the longest on average time it takes to get to work?

Group By

```
/* Group By */  
SELECT State,  
Rates_Property_Theft,  
Totals_Violent_Robbery,  
Population  
FROM state_crime  
WHERE Totals_Violent_Robbery > 10000  
GROUP BY Totals_Violent_Robbery  
ORDER BY Totals_Violent_Robbery DESC;
```

Detailed view

```
SELECT column_name(s)  
FROM table_name  
WHERE condition  
GROUP BY column_name(s)  
ORDER BY column_name(s);
```

What do you want to see?

```
SELECT State,  
Rates_Property_Theft,  
Totals_Violent_Robbery,  
Population
```

FROM state_crime → Table name

WHERE Totals_Violent_Robbery > 10000 → Column and Condition

GROUP BY Totals_Violent_Robbery

order by Totals_Violent_Robbery DESC; → syntax

Column names

Clause: to filter what table?

Puts rows that are similar together

Sort results in ascending or descending order

Column to sort by

File Edit View Navigate Code Refactor Run Tools Git Window Help QCLWorkshop - console_2

Database Consoles > @localhost > console_2

Database

@localhost 2 of 6

information_schema

qcworkshop

tables 2

state_computer_data

columns 4

State varchar(20)
Persons_per_Household float
Households_with_computer float
Households_with_Internet float

state_crime

columns 7

State text
Crime_Year int
Population int
Rates_Property_Theft double
Rates_Violent_Robbery double
Totals_Property_Theft int
Totals_Violent_Robbery int

Server Objects

console_2 x state_computer_data_insert.sql x qcworkshop console_2

33 SELECT State,
34 Population,
35 Totals_Property_Theft,
36 Totals_Violent_Robbery,
37 Crime_Year
38 FROM state_crime
39 WHERE Totals_Violent_Robbery >=3000
40 ORDER BY Population DESC;

41 /* Group By */

42 SELECT State,
43 Rates_Property_Theft,
44 Totals_Violent_Robbery,
45 Population
46 FROM state_crime
47 WHERE Totals_Violent_Robbery > 10000
48 GROUP BY Totals_Violent_Robbery
49 ORDER BY Totals_Violent_Robbery DESC;

50

51

Services

Tx

@localhost default 53 ms
console_2 54 ms
console_2 54 ms
state_computer_data_insert.sql

Output x Group By x

19 rows > 19 rows < Tx: Auto DDL CSV

	State	Rates_Property_Theft	Totals_Violent_Robbery	Population
1	United States	701	369089	309330219
2	United States	340.5	267988	328239523
3	United States	2596.1	84041	318857056
4	California	612.9	58116	37338198
5	California	386.1	52301	39512223
6	Texas	905.2	32843	25253466
7	Texas	392.8	28988	28995881
8	New York	339.5	28630	19395206

Event Log

19 rows retrieved starting from 1 in 47 ms (execution: 2 ms, fetching: 45 ms)

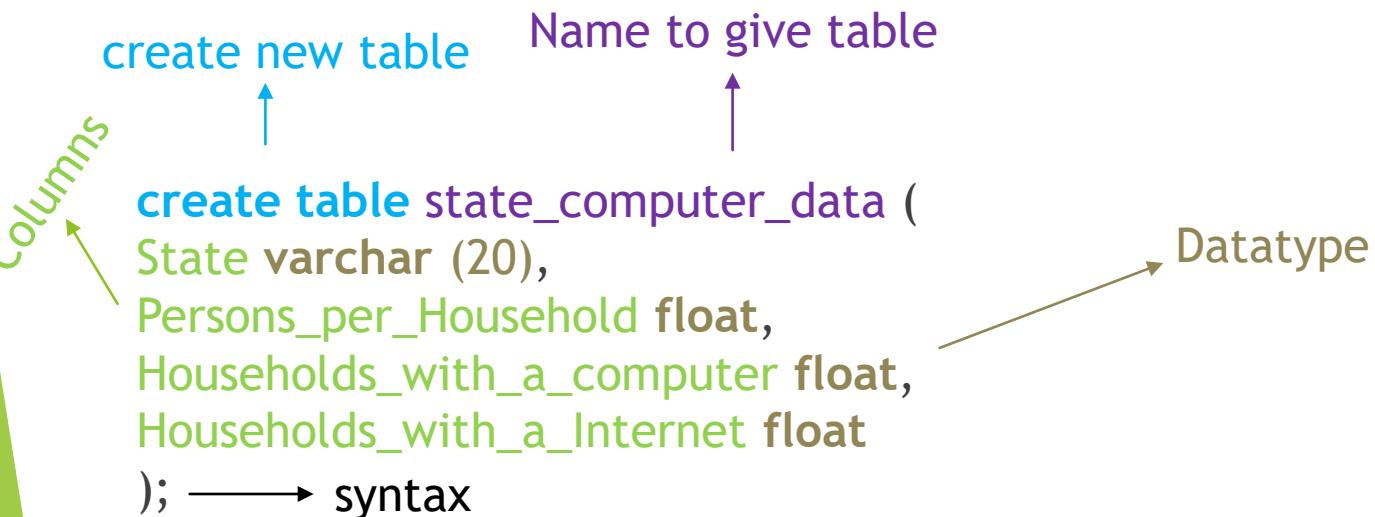
42:1 (209 chars, 8 line breaks) CRLF UTF-8 4 spaces

Create Table

```
/* Create Table */  
CREATE TABLE state_computer_data(  
State VARCHAR (20),  
Persons_per_Household FLOAT,  
Households_with_computer FLOAT,  
Households_with_Internet FLOAT  
);
```

Detailed view

```
CREATE TABLE table_name (  
    column1 datatype,  
    column2 datatype,  
    column3 datatype,  
    ....  
)
```

create new table Name to give table

create table state_computer_data (
State varchar (20),
Persons_per_Household float,
Households_with_a_computer float,
Households_with_a_Internet float
); —→ syntax

The screenshot shows a DataGrip IDE interface with a database console session titled "console_2".

Database Explorer: Shows the database structure at `@localhost`. It includes the `information_schema`, the `qdworkshop` schema (which contains a `state_computer_data` table and a `state_crime` table), and `Server Objects`.

Console: Displays the following SQL query and its execution results.

```
44 | Rates_Property_Theft,
45 | Totals_Violent_Robbery,
46 | Population
47 | FROM state_crime
48 | WHERE Totals_Violent_Robbery > 10000
49 | GROUP BY State
50 | ORDER BY Totals_Violent_Robbery DESC;
51 |
52 /* Create Table */
53 ✓ CREATE TABLE state_computer_data(
54     State VARCHAR (20),
55     Persons_per_Household FLOAT,
56     Households_with_computer FLOAT,
57     Households_with_Internet FLOAT
58 );
```

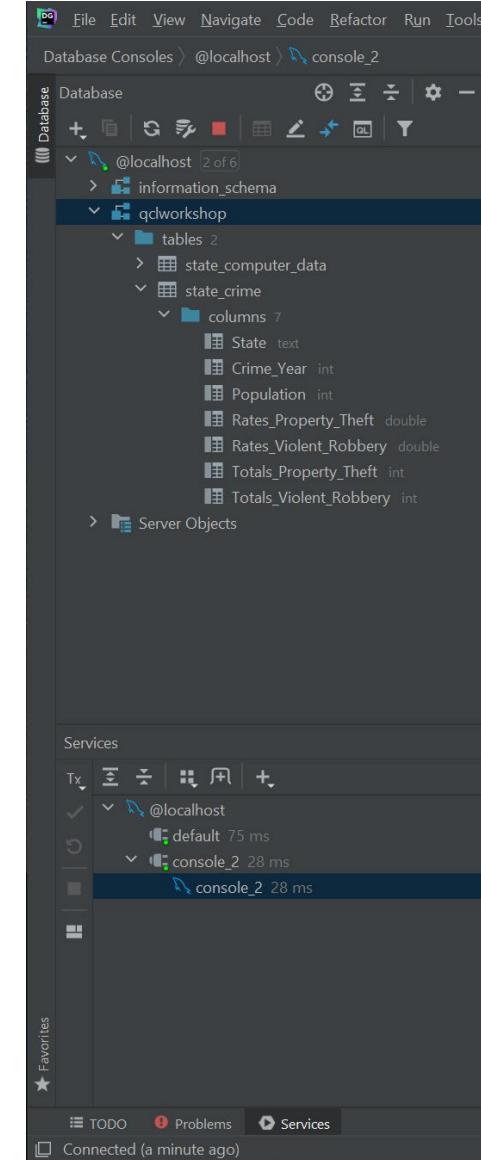
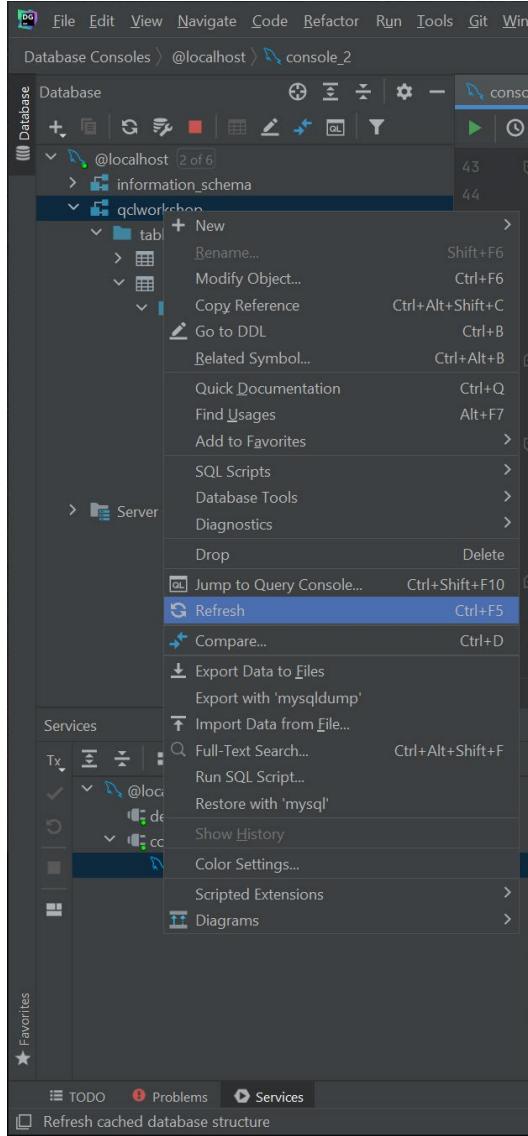
Services: Shows the transaction log with entries for `@localhost` and the current session `console_2`.

Output: Shows the execution results of the query and the creation of the `state_computer_data` table.

```
ORDER BY Totals_Violent_Robbery DESC
[2021-11-23 16:46:38] 12 rows retrieved starting from 1 in 58 ms (execution: 5 ms, fetching: 53 ms)
qcworkshop> CREATE TABLE state_computer_data(
    State VARCHAR (20),
    Persons_per_Household FLOAT,
    Households_with_computer FLOAT,
    Households_with_Internet FLOAT
)
[2021-11-23 16:46:59] completed in 13 ms
```

Status Bar: Shows the connection status, character count (52:1 (167 chars, 6 line breaks)), file encoding (CRLF), and line endings (UTF-8).

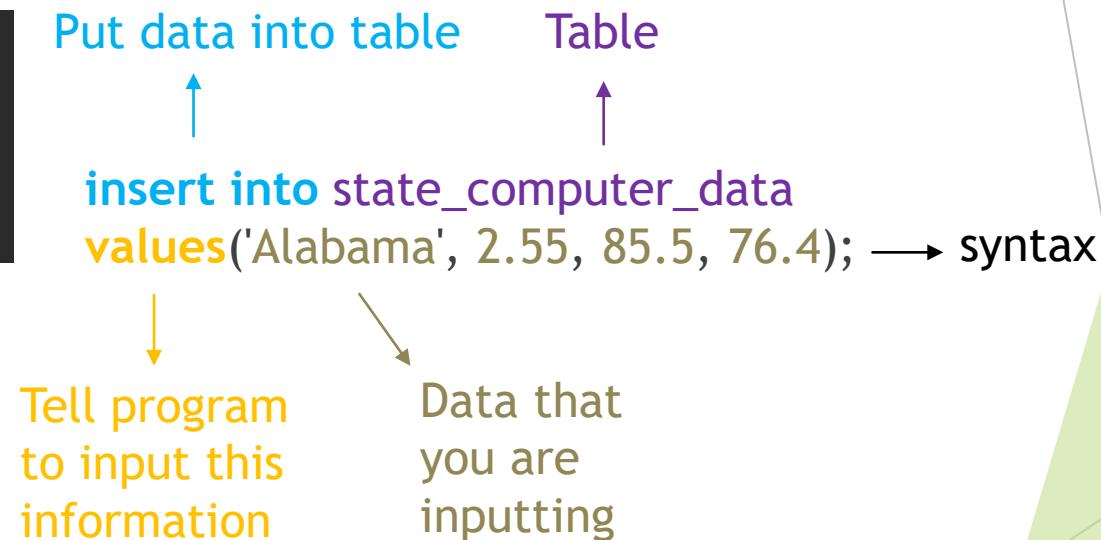
Refresh



Insert and Detail View

```
INSERT INTO table_name (column1, column2, column3, ...)  
VALUES (value1, value2, value3, ...);
```

```
/* Insert into */  
INSERT INTO state_computer_data  
VALUES('Alabama', 2.55, 85.5, 76.4);
```



The screenshot shows the DataGrip IDE interface with a database query being run in a console.

Database Explorer: Shows the database structure at `@localhost`, including the `qclworkshop` schema which contains the `state_computer_data` and `state_crime` tables. The `state_crime` table has 7 columns: `State`, `Crime_Year`, `Population`, `Rates_Property_Theft`, `Rates_Violent_Robbery`, `Totals_Property_Theft`, and `Totals_Violent_Robbery`.

Console: The query being run is:

```
46    Population
47    FROM state_crime
48    WHERE Totals_Violent_Robbery > 10000
49    GROUP BY State
50    ORDER BY Totals_Violent_Robbery DESC;
51
52    /* Create Table */
53    CREATE TABLE state_computer_data(
54        State VARCHAR (20),
55        Persons_per_Household FLOAT,
56        Households_with_computer FLOAT,
57        Households_with_Internet FLOAT
58    );
59
60    /* Insert into */
61    INSERT INTO state_computer_data
62        VALUES('Alabama', 2.55, 85.5, 76.4);
```

Services: Shows the transaction status for the current session, which is running a query in the `console_2` service.

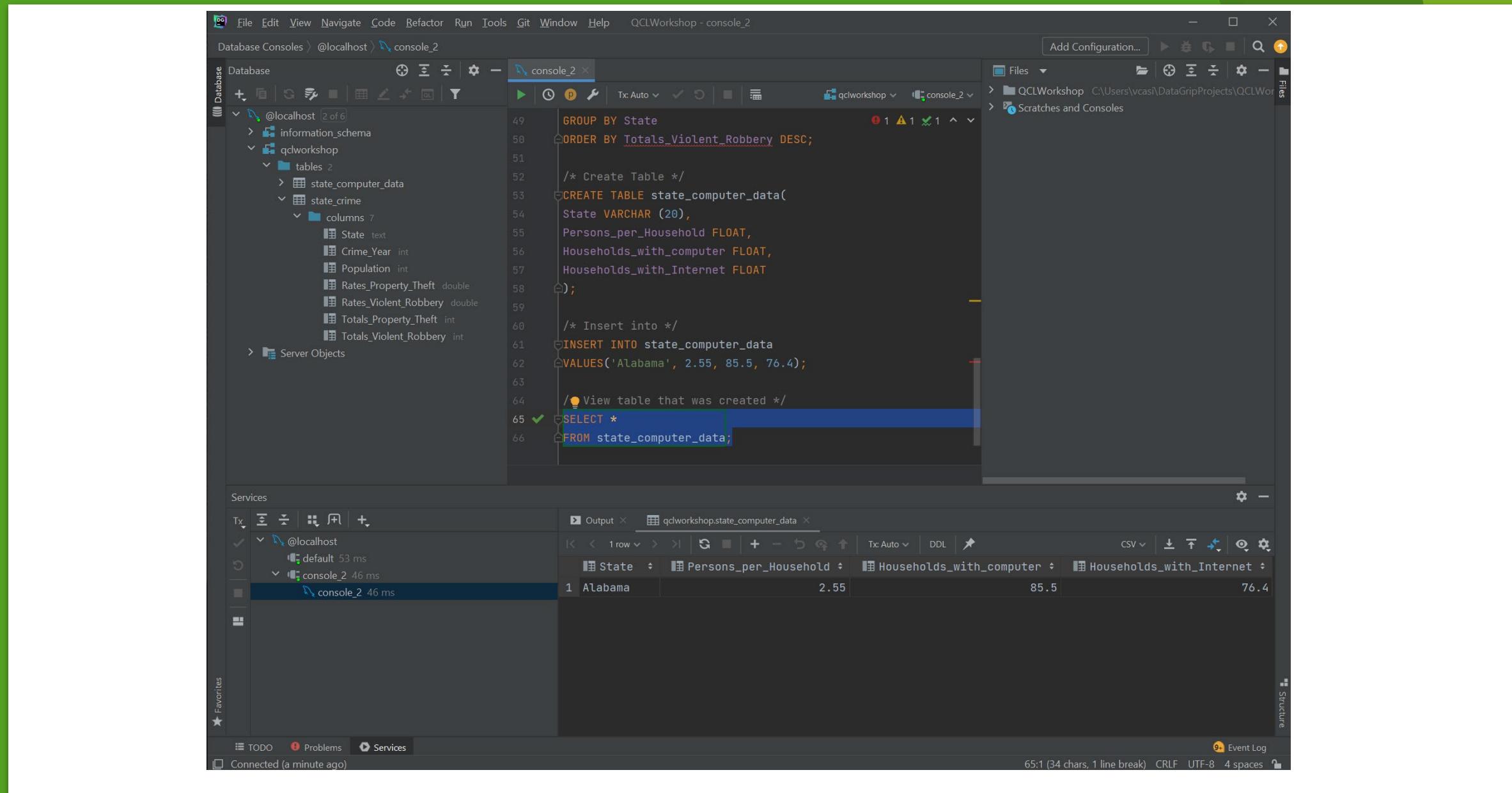
Output: Displays the results of the query execution:

```
State VARCHAR (20),
Persons_per_Household FLOAT,
Households_with_computer FLOAT,
Households_with_Internet FLOAT
)
[2021-11-23 16:46:59] completed in 13 ms
qclworkshop> INSERT INTO state_computer_data
VALUES('Alabama', 2.55, 85.5, 76.4)
[2021-11-23 16:47:33] 1 row affected in 5 ms
```

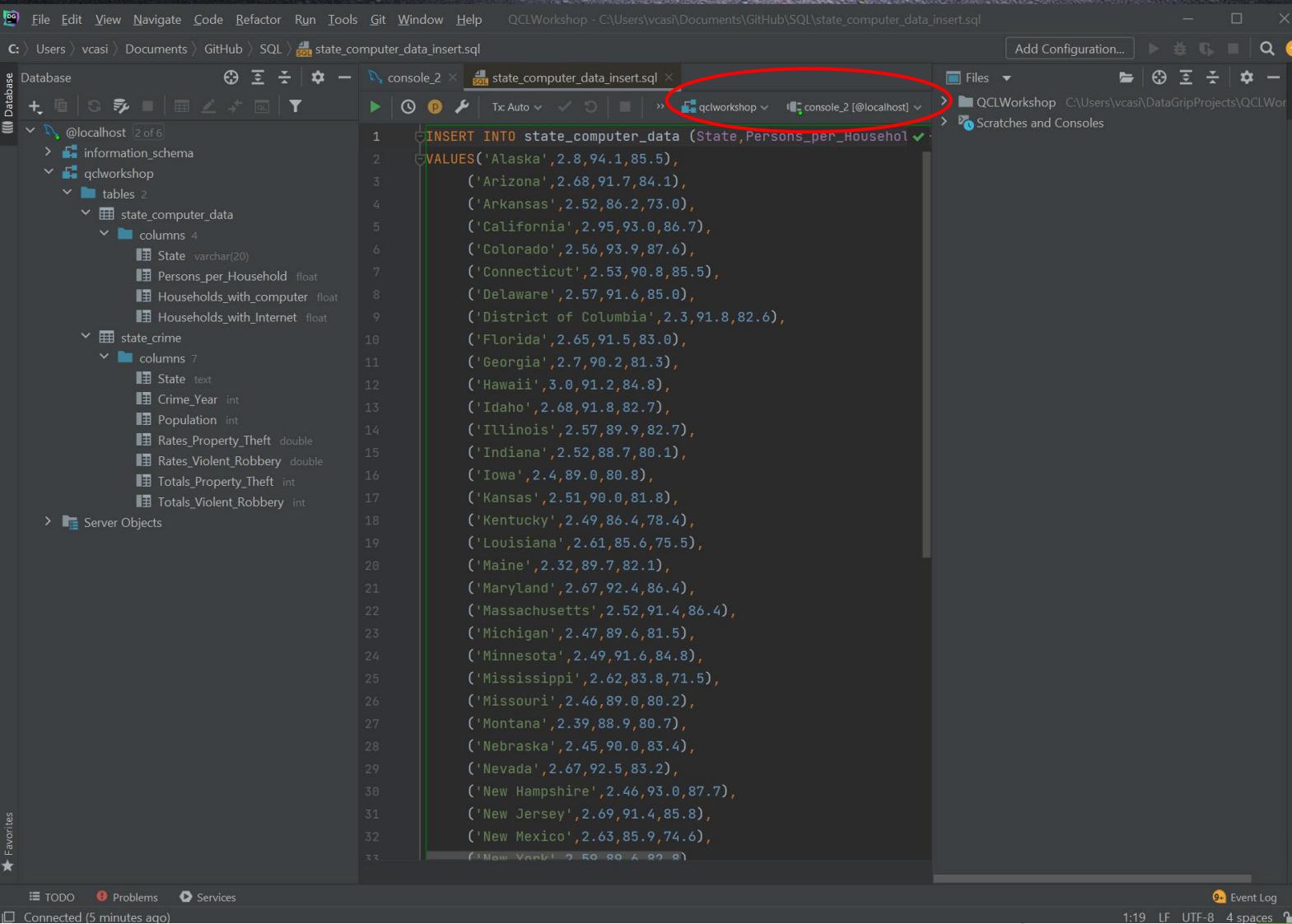
Bottom Status Bar: Shows the following information: 1 row affected in 5 ms, Event Log, 60:1 (86 chars, 2 line breaks), CRLF, UTF-8, 4 spaces.

Look at table with one column

```
/* View table that was created */  
SELECT *  
FROM state_computer_data;
```



Insert data (alternative)



The screenshot shows the DataGrip IDE interface with the following details:

- File Path:** C:\Users\vcasi\Documents\GitHub\SQL\state_computer_data_insert.sql
- Database:** Database tool window showing the schema structure of the 'qdworkshop' database, including tables like 'state_computer_data' and 'state_crime'.
- SQL Editor:** The main editor window contains the following SQL code:

```
INSERT INTO state_computer_data (State, Persons_per_Household, Households_with_computer, Households_with_Internet)
VALUES('Alaska', 2.8, 94.1, 85.5),
      ('Arizona', 2.68, 91.7, 84.1),
      ('Arkansas', 2.52, 86.2, 73.0),
      ('California', 2.95, 93.0, 86.7),
      ('Colorado', 2.56, 93.9, 87.6),
      ('Connecticut', 2.53, 90.8, 85.5),
      ('Delaware', 2.57, 91.6, 85.0),
      ('District of Columbia', 2.3, 91.8, 82.6),
      ('Florida', 2.65, 91.5, 83.0),
      ('Georgia', 2.7, 90.2, 81.3),
      ('Hawaii', 3.0, 91.2, 84.8),
      ('Idaho', 2.68, 91.8, 82.7),
      ('Illinois', 2.57, 89.9, 82.7),
      ('Indiana', 2.52, 88.7, 80.1),
      ('Iowa', 2.4, 89.0, 80.8),
      ('Kansas', 2.51, 90.0, 81.8),
      ('Kentucky', 2.49, 86.4, 78.4),
      ('Louisiana', 2.61, 85.6, 75.5),
      ('Maine', 2.32, 89.7, 82.1),
      ('Maryland', 2.67, 92.4, 86.4),
      ('Massachusetts', 2.52, 91.4, 86.4),
      ('Michigan', 2.47, 89.6, 81.5),
      ('Minnesota', 2.49, 91.6, 84.8),
      ('Mississippi', 2.62, 83.8, 71.5),
      ('Missouri', 2.46, 89.0, 80.2),
      ('Montana', 2.39, 88.9, 80.7),
      ('Nebraska', 2.45, 90.0, 83.4),
      ('Nevada', 2.67, 92.5, 83.2),
      ('New Hampshire', 2.46, 93.0, 87.7),
      ('New Jersey', 2.69, 91.4, 85.8),
      ('New Mexico', 2.63, 85.9, 74.6),
      ('New York', 2.59, 90.4, 82.9)
```
- Toolbars and Menus:** Standard DataGrip menu bar (File, Edit, View, Navigate, Code, Refactor, Run, Tools, Git, Window, Help) and toolbar buttons.
- Status Bar:** Shows the connection status ("Connected (5 minutes ago)", "Event Log", "1:19", "LF", "UTF-8", "4 spaces").

File Edit View Navigate Code Refactor Run Tools Git Window Help QCLWorkshop - C:\Users\vcasi\Documents\GitHub\SQL\state_computer_data_insert.sql

C: > Users > vcasi > Documents > GitHub > SQL > state_computer_data_insert.sql

Database Database + ⊖ ⊖ ⊖ | Settings

@localhost [2 of 6]

- > information_schema
- > qcworkshop
 - tables 2
 - state_computer_data
 - columns 4
 - State varchar(20)
 - Persons_per_Household float
 - Households_with_computer float
 - Households_with_Internet float
 - state_crime
 - columns 7
 - State text
 - Crime_Year int
 - Population int
 - Rates_Property_Theft double
 - Rates_Violent_Robbery double
 - Totals_Property_Theft int
 - Totals_Violent_Robbery int
- > Server Objects

Services Services Tx ⊖ ⊖ | F | +

@localhost

- default 53 ms
- console_2 16 ms
 - console_2 16 ms

state_computer_data_insert.sql 16 ms

Add Configuration... | Tx Auto | Structure | Event Log

50 rows affected in 7 ms [2021-11-23 16:53:13] 50 rows affected in 7 ms

Files Files | ⊖ ⊖ | Settings

QCLWorkshop C:\Users\vcasi\Documents\GitHub\SQL\state_computer_data_insert.sql

Scratches and Consoles

25:37 (1694 chars, 50 line breaks) LF UTF-8 4 spaces

Insert data (cont.)

Look at table data

Which column looks like a column that has the same information for both tables?

```
/* View table that was created */
SELECT *
FROM state_computer_data;
```

The screenshot shows the DataGrip IDE interface with the following details:

- Database Explorer:** Shows the database structure. Under the `@localhost` connection, there is a `state_computer_data` table with four columns: `State` (varchar(20)), `Persons_per_Household` (float), `Households_with_computer` (float), and `Households_with_Internet` (float).
- SQL Editor:** Contains the following SQL code:

```
Households_with_Internet FLOAT
);
/* Insert into */
INSERT INTO state_computer_data
VALUES('Alabama', 2.55, 85.5, 76.4);

/* View table that was created */
SELECT *
FROM state_computer_data;

/* Alternative view of Insert into */
SELECT *
FROM state_computer_data;
```
- Services:** Shows a transaction named `console_2` with a duration of 42 ms.
- Output Tab:** Displays the results of the `SELECT` query from the SQL editor, showing data for eight states:

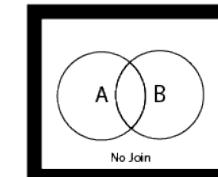
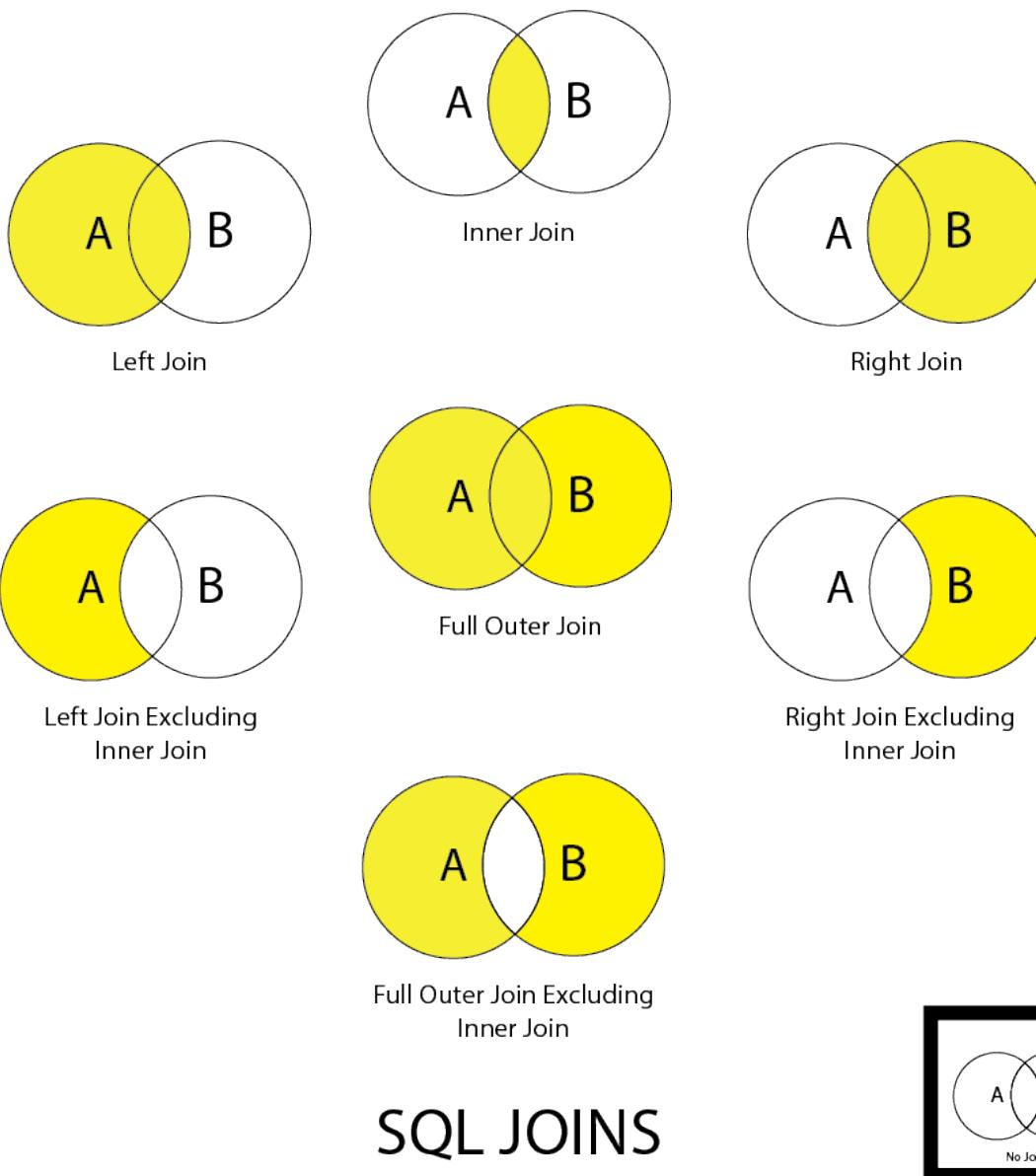
State	Persons_per_Household	Households_with_computer	Households_with_Internet
Alabama	2.55	85.5	76.4
Alaska	2.8	94.1	86.2
Arizona	2.68	91.7	86.2
Arkansas	2.52	93	86.2
California	2.95	93.9	86.2
Colorado	2.56	90.8	86.2
Connecticut	2.53	91.6	86.2
Delaware	2.57	91.6	86.2

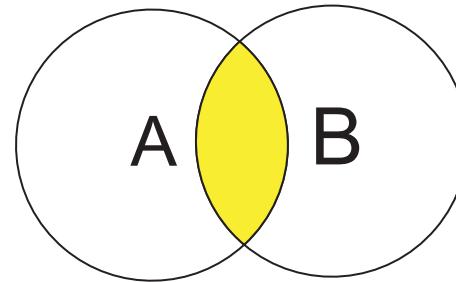
At the bottom, status bar text includes: Connected (8 minutes ago), 69:1 (34 chars, 1 line break), CRLF, UTF-8, 4 spaces.



Activity #3 - Create Table and Insert Data

- ▶ Create a table named state_people and add the attributes: State as a varchar with 20 characters, Employment_Firms_Total as a integer, Age_Percent_Under_18_Years as a float, and Age_Percent_65_and_Older as a float.
- ▶ Insert data into your new table called state_people using a sql file in the downloaded github folder





Inner Join

```
/* Inner Join */  
SELECT state_computer_data.State,  
state_computer_data.Households_with_computer,  
state_crime.Totals_Property_Theft  
FROM state_computer_data  
JOIN state_crime  
ON state_computer_data.state = state_crime.State;
```

Detail View

```
SELECT column_name(s)
FROM table1
INNER JOIN table2
ON table1.column_name = table2.column_name;
```

what table?

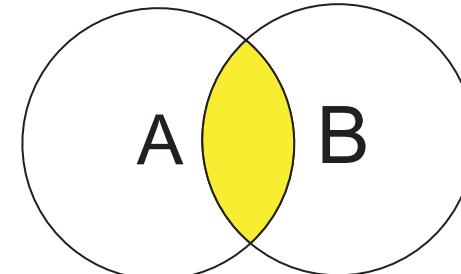
What do you
want to see?

Table name Column names

```
SELECT state_computer_data.State,
state_computer_data.Households_with_a_computer,
state_crime.Totals_Property_Theft
← FROM state_computer_data
JOIN state_crime
ON state_computer_data.state = state_crime.State; → syntax
```

What table are you joining and on what columns?

You can just join if you are making an inner join, yet if you are
making any other join, you do have to specify



Inner Join

File Edit View Navigate Code Refactor Run Tools Git Window Help QCLWorkshop - console_2

Database Consoles > @localhost > console_2

Database

@localhost [2 of 6]

- information_schema
- qdworkshop
 - tables 2
 - state_computer_data
 - columns 4
 - State varchar(20)
 - Persons_per_Household float
 - Households_with_computer float
 - Households_with_Internet float
 - state_crime
 - columns 7
 - State text
 - Crime_Year int
 - Population int
 - Rates_Property_Theft double
 - Rates_Violent_Robbery double
 - Totals_Property_Theft int
 - Totals_Violent_Robbery int
 - Server Objects

console_2 × state_computer_data_insert.sql

```
VALUES('Alabama', 2.55, 85.5, 76.4);
/* View table that was created */
SELECT *
FROM state_computer_data;
/* Alternative view of Insert into */
SELECT *
FROM state_computer_data;
/* Inner Join */
SELECT state_computer_data.State,
       state_computer_data.Households_with_computer,
       state_crime.Totals_Property_Theft
  FROM state_computer_data
 JOIN state_crime
    ON state_computer_data.state = state_crime.State;
```

Services

@localhost

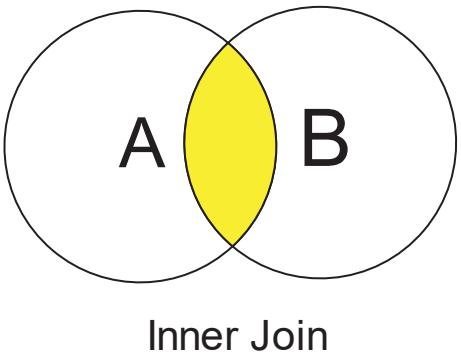
 - default 53 ms
 - console_2 53 ms
 - console_2 53 ms
 - state_computer_data_insert.sql

Output × Result 17 ×

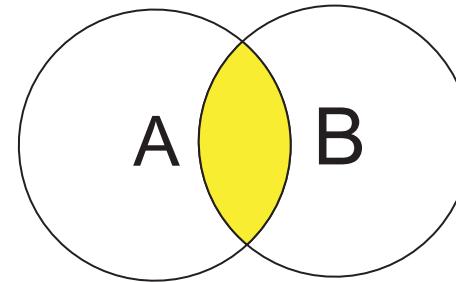
State	Households_with_computer	Totals_Property_Theft
1 Alabama	85.5	26079
2 Alaska	94.1	3563
3 Arizona	91.7	28699
4 Arkansas	86.2	18695
5 California	93	152555
6 Colorado	93.9	20064
7 Connecticut	90.8	6441
8 Delaware	91.6	2968

153 rows retrieved starting from 1 in 43 ms (execution: 3 ms, fetching: 40 ms)

73:1 (205 chars, 5 line breaks) CRLF UTF-8 4 spaces



Alias



Inner Join

```
/* Alias */  
SELECT a.State,  
a.Households_with_computer,  
b.Totals_Property_Theft,  
b.Totals_Violent_Robbery  
FROM state_computer_data AS a  
JOIN state_crime AS b  
ON a.state = b.State;
```

Detail View

```
SELECT column_name(s)  
FROM table_name AS alias_name;
```

what table?

What do you
want to see?

Table name
↑
`SELECT a.State,`

`a.Households_with_a_computer,`
`b.Totals_Property_Theft,`
`b.Totals_Violent_Robbery`

Column names

↑

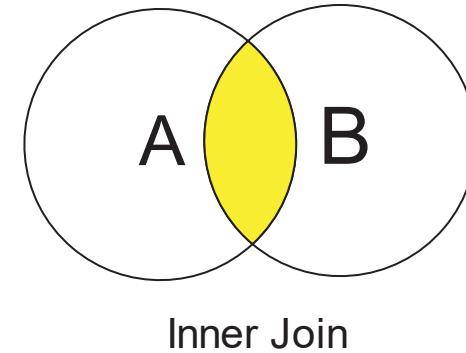
`FROM state_computer_data AS a`

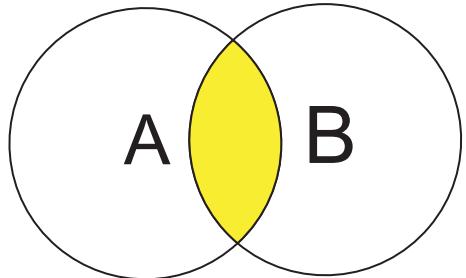
`JOIN state_crime AS b`

`ON a.state = b.State;` → syntax

↓
What table are you joining and on what columns?

You can just join if you are making an inner join, yet if you are
making any other join, you do have to specify





Inner Join

File Edit View Navigate Code Refactor Run Tools Git Window Help QCLWorkshop - console 2

Database Consoles > @localhost > console_2

Database Services

console_2 x state_computer_data_insert.sql

```
FROM state_computer_data;
/* Inner Join */
SELECT state_computer_data.State,
state_computer_data.Households_with_computer,
state_crime.Totals_Property_Theft
FROM state_computer_data
JOIN state_crime
ON state_computer_data.state = state_crime.State;
/* Alias */
SELECT a.State,
a.Households_with_computer,
b.Totals_Property_Theft,
b.Totals_Violent_Robbery
FROM state_computer_data AS a
JOIN state_crime AS b
ON a.state = b.State;
```

Output x Alias x

State	Households_with_computer	Totals_Property_Theft	Totals_Violent_Robbery
Alabama	85.5	26079	
Alaska	94.1	3563	
Arizona	91.7	28699	
Arkansas	86.2	18095	
California	93	152555	
Colorado	93.9	20064	
Connecticut	90.8	6441	
Delaware	91.6	2968	

Event Log

Connected (15 minutes ago)



Activity #4 - Inner Join with Alias

- ▶ Create an inner join using aliases with tables state_workforce and state_people, make sure to view attributes: state, Mean_Travel_Time_to_Work, and Employment_Firms_Total

Writing Query

Select - Returns the data that was requested

From - choose a table to draw information from

Join - matches records from different tables

Where - filters data bases on request

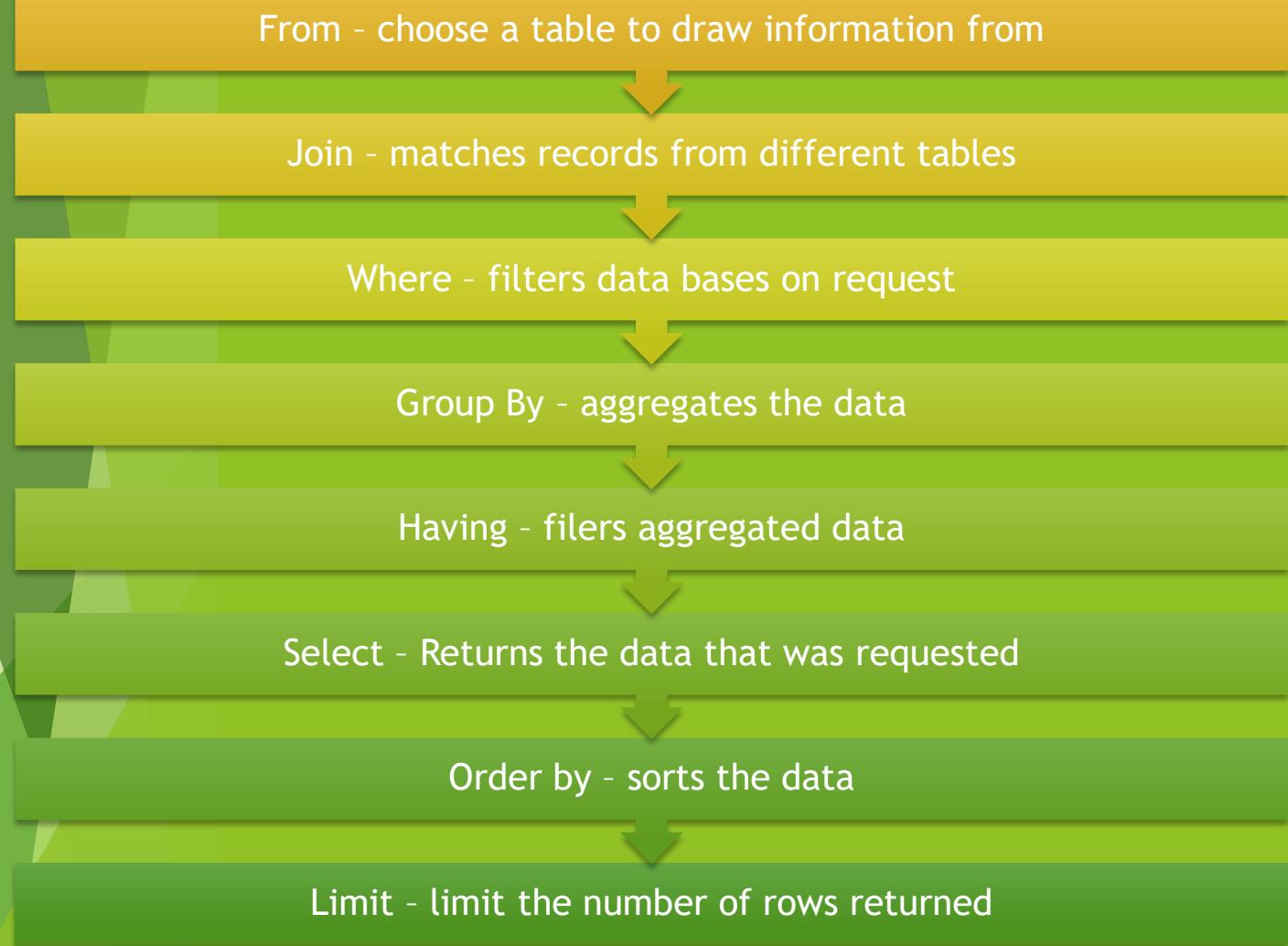
Group By - aggregates the data

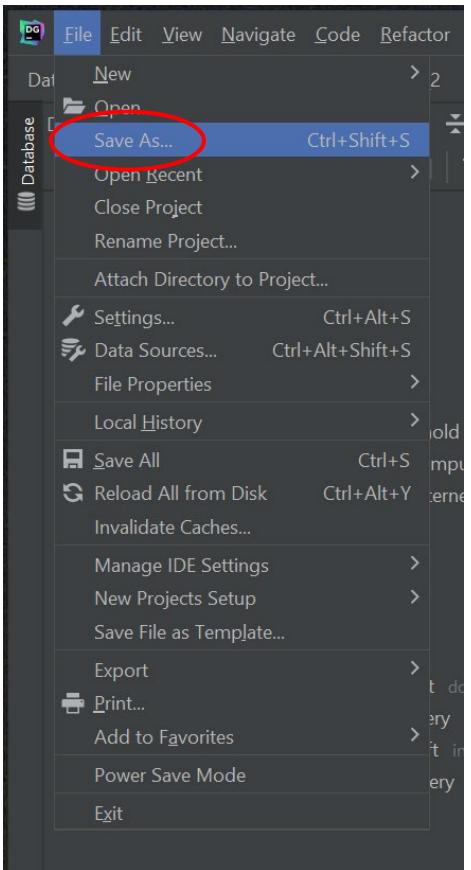
Having - filers aggregated data

Order by - sorts the data

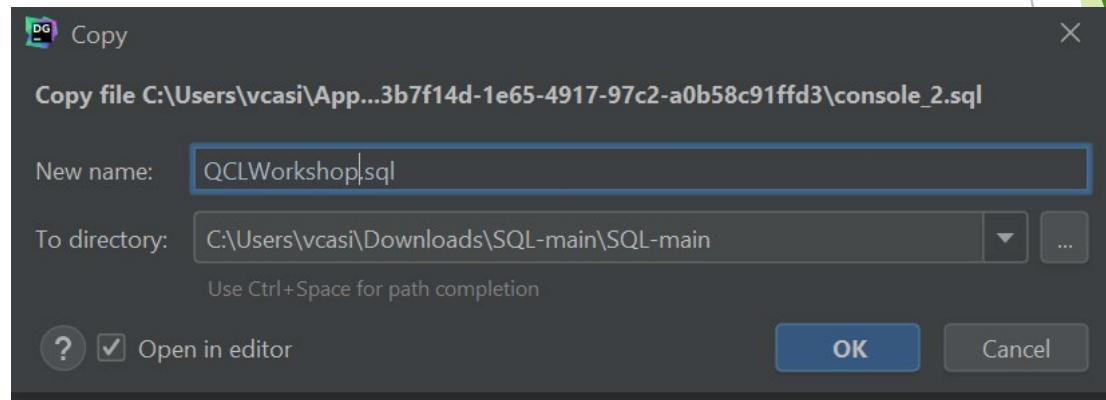
Limit - limit the number of rows returned

Operation Order





Save



Resources

- ▶ Dbeaver Wiki - <https://github.com/dbeaver/dbeaver/wiki>
- ▶ W3schools - <https://www.w3schools.com/sql/default.asp>
- ▶ Code Academy- <https://www.w3schools.com/sql/default.asp>
- ▶ Quizlet - <https://quizlet.com/> (for vocab testing)

Best way to learn

- ▶ SQL Murder Mystery - <https://mystery.knightlab.com/>

Contact info

- ▶ QCL: QCL@cmc.edu
- ▶ Vanessa Casillas: vanessa.casillas@claremontmckenna.edu